Published in:
IEEE Communications Magazine
September 2022

Hybrid Edge Cloud: A pragmatic approach for decentralized cloud computing©

reimagining the cloud

10.1109/MCOM.001.2200251 © 2022 IEEE Published in
IEEE Communications Magazine September 2022

mimik

DOCUMENT NAME

Hybrid Edge Cloud: A Pragmatic Approach for Decentralized Cloud Computing.

DATE

2022/07/30

AUTHORS

Siavash M. Alamouti,
Fay Arjomandi,
Michel Burger

ABSTRACT

The massive growth of connected devices including sensors and machines is revolutionizing every aspect of human life. The socioeconomic impacts are significant and have already transformed many industries. In this paper, we discuss some of the challenges of the explosion of devices and exponential growth in personal digital content and machine generated data.

The centralized cloud architecture adopted in the early days of mobile internet was designed primarily to allow access to data stored on the world wide web. Today, use cases have evolved significantly. Humans and devices are now producing most of the data consumed on the internet. As a result, the existing centralized cloud infrastructure is no longer efficient nor sustainable.

There is significant waste of network bandwidth to send terabytes of data to server farms that may be hundreds of kilometers away from the source and/or destination of data. The prevalent centralized cloud architecture does not adequately leverage massive amounts of computing resources on smart devices which are idle most of the time.

Moreover, despite all the efforts to reduce network latencies, communication to server farms is a major bottleneck for latency in many applications. We introduce a novel architectural approach to cloud decentralization called Hybrid Edge Cloud (HEC) that minimizes network bandwidth usage, reduces communication latencies, and leverages resources on smart devices to reduce the burden on server farms and other centralized computing resources.

HEC combines the benefits of new network technologies such as 5G and WiFi 6 in private and public clouds to leverage computing resources on smart devices to build a sustainable decentralized infrastructure for the hyper-connected world.

10.1109/MCOM.001.2200251 © 2022 IEEE Published in
IEEE Communications Magazine September 2022

10.1109/MCOM.001.2200251 © 2022 IEEE Published in
IEEE Communications Magazine September 2022

1.INTRODUCTION

The first four generations of cellular technologies have changed the landscape of mobile communications by changing the concept of telephony from calling to and from places to individuals [1] . The second and third generations enabled pervasive text communication and web browsing. Eventually, 4G and Wi-Fi provided ubiquitous access to the Internet and ushered in the era of mobile Internet and the app economy. In the early days of mobile Internet, the primary usage of these networks was for smartphones and later Internet of Things (IoT) devices to get access to information and apps hosted on the worldwide web. Initially, people and machines mostly acted as consumers of the web content. As a result, the centralized cloud architecture was designed with the key assumption that most data would reside on centralized servers across the worldwide web, and client nodes would primarily be used to access these centralized sources of data and information. This assumption no longer holds true. The use of mobile Internet has evolved, and the roles of people and machines have transformed from consumers to producers of data. Most of the data are now introduced by smart devices at the edge of the network [2] [3] [4] .

The "cloud" [5] has been a major enabler for mobile internet. Today, the most popular consumer and enterprise applications and solutions are hosted in data centers. This is what we refer to as the "cloud" which has been essential for enabling the app economy. The underlying solution architecture is a hierarchical client-server architecture. Certain nodes act as servers and others act as clients. By contrast, in a peer-to-peer architecture, any node can act as both a server and client. Most computing nodes operate in a client-server mode, where most servers are in data centers made up of server farms scattered around the world. This fixed and hierarchical client-server architecture may have been efficient for hosting applications that provide access to content and information from remote servers to billions of "client" devices. Solutions' backends are hosted on servers that handle compute-intensive tasks and the client application software are run on devices such as smart phones that perform simpler functions such as entering commands, caching content, and rendering information for the end user.

The latest evolution of cloud architecture is the move to microservices [6] which decomposes a monolithic backend solution into a collection of microservices that are dynamically instantiated (serverless[1]) behind an API gateway. This evolution introduced new complexities in microservice-to-microservice communication (service mesh [7]) and cluster management [8] .

The move to microservices is triggered by three major trends:

- API: Microservices implement and expose RESTful APIs (HTTP REST-based). A set of easy-to-use APIs can hide internal complexities and facilitate communication between the microservices within a system.
- Automated deployment: It is possible to build complex systems with a potentially large number of elements by deploying microservices automatically using deployment scripts (e.g., Ansible) controlled by a pipeline infrastructure (e.g.,

[1] The term serverless can be confusing as it doesn't mean that servers are not required. Instead, it means that the server role is not permanently assigned [29] .

10.1109/MCOM.001.2200251 © 2022 IEEE Published in IEEE Communications Magazine September 2022

Jenkins). Moreover, automated deployment can help build flexible systems by providing the ability to determine where deployments occur.

- On-demand IT resources: The ability to request IT resources (CPU, storage, and network) through simple APIs and to obtain these resources in a near real-time fashion makes the creation of large and scalable systems more feasible.

Figure 1 illustrates the typical deployment of an application in a centralized architecture. A client application runs on a smart device, and a collection of central cloud[2] functions host the backend of the solution. This backend is usually made up of microservices (μs) reachable through an API gateway. As shown, every http request is sent from the "client" device to servers in the central cloud.

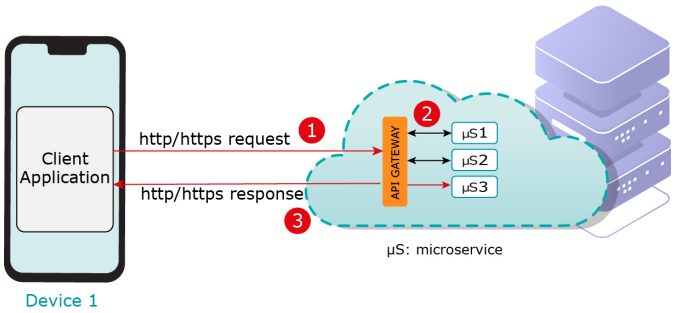

Figure 1. Fixed client-server architecture using centralized cloud.

A special case of device-to-device communication is illustrated in **Figure 2**.

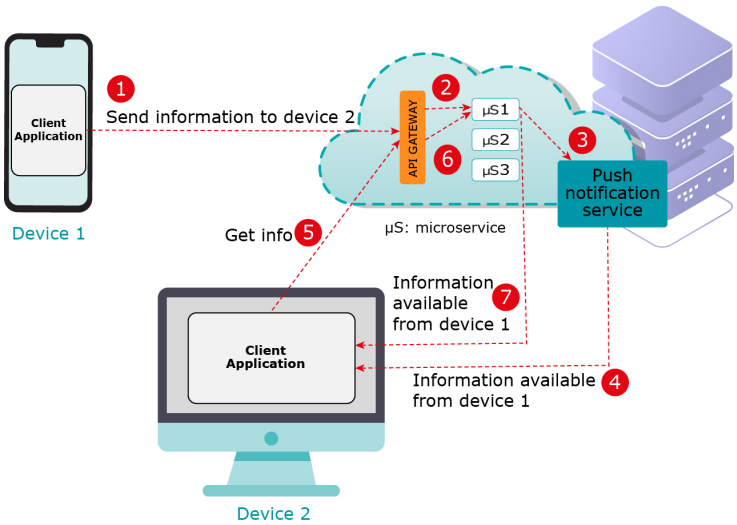

Figure 2. Device-to-device communications in fixed client-server architecture of centralized cloud.

[2] Hereon, we refer to server farms in data centers as central cloud.

In this case, when device 1 wishes to send information to device 2, it sends an http request that ends up at the API gateway, and the appropriate microservice is launched. Usually, a push notification service informs device 2 that information is available, and device 2 responds with a request that is serviced again by a microservice hosted on the central cloud. In other words, even if the two client devices are in proximity and on the same local network, all communications and data need to go through servers in a data center that may be 100s kilometers away! The same holds true, even for two applications running on the same smart device.

The major advantage of this architecture is the rapid and low-cost deployment of (computing-and/or storage-intensive) applications on generic servers shared amongst many applications with the aid of virtualization and orchestration technologies [5] . However, within the last decade, we have witnessed three fundamental trends that make a hierarchical client-server architecture less efficient. The first is the explosion of computing devices and embedded computing in all things [9] , and the increasing capabilities of smart devices. For instance, there are more computing, memory, and storage available in today's smartphones than in powerful servers just a decade ago. This trend will continue owing to Moore's law [10] . The second trend is the enormous amount of data generated on these (edge) devices. With the advent of social media on mobile devices, orders of magnitude more personal multimedia content is generated on devices (photos, videos, sensor data, etc.) rather than premium content from major studios and broadcasters hosted on central servers in the cloud [11] [12] [13] . Currently, most of the data generated on smart devices are sent back to the central cloud for processing and to facilitate sharing. The third trend is the decomposition of solutions in the collection of microservices and the automation of deployment that make backend solutions much more dynamic (serverless) with scalability that closely fits the demand either in volume or even geography.

The current hierarchical architecture makes central cloud resources and network connectivity bottlenecks for future growth. Sending data from hundreds of billions of client devices to thousands of centralized cloud servers can waste bandwidth and energy, which can have serious social and economic implications.

Another disadvantage of centralized cloud architecture is developers' reliance on cloud service providers who have access to the apps and the data stored or processed in their servers [14] [15] [16] . As a result, a handful of very large companies have control over the vast majority of consumer and enterprise data. In addition, despite all sophisticated security measures, storing data and hosting applications on third-party resources expose the owners of the information to risks. Cloud resources have been designed for easy access to millions of developers and application service providers, which has increased vulnerabilities and security holes. This has resulted in gross abuse of consumer and enterprise data privacy and security [17] [18] [19] [20] [21] [22] .

10.1109/MCOM.001.2200251 © 2022 IEEE Published in
IEEE Communications Magazine September 2022

An effective and feasible approach to address this dilemma is to enable any smart device to act as a cloud server when it makes sense. Enabling smart devices to act as cloud servers can potentially reduce the reliance on third-party cloud services that are not necessary for applications and allow microservice-based solutions to be more flexible by dynamically moving microservices from the backend to smart devices.

Many of the functions performed in the central cloud can then be performed on smart devices that act as servers. In effect, we can create a physical decentralized cloud fabric that is potentially several orders of magnitude larger than the central cloud fabric. For example, there are currently over 80 million Sony PlayStation 4 (PS4) consoles in people's homes [23] . This represents more than 600 million processor cores and 40,000 petabytes of storage. In comparison, this is much larger computing, storage, and memory resources in the aggregate than the entire Amazon Web Services (AWS) infrastructure [24] . The PS4 is only one type of device. Billions of PCs, set-top-boxes, game consoles, streaming players, routers, smart phones, tablets, and other computing devices can potentially act as cloud servers and collectively have orders of magnitude more computing power than the central cloud.

Therefore, we have the opportunity to create a cloud fabric composed of tens of billions of smart devices that are currently being used as clients only. Once we enable these devices to act as servers, we can build a decentralized cloud that is orders of magnitude larger than the current centralized cloud.

The benefits of such an architecture are phenomenal: reduced cloud hosting costs, reduced communication bandwidth and network efficiency, reduced energy consumption and carbon emissions, reduced latency, reduced application development time, accelerating the microservice trend, increasing data privacy, and providing consumers and enterprises better control over their data.

In this paper, we discuss the opportunities and challenges for cloud decentralization and some of the drivers, requirements, and principles necessary to establish a pragmatic and scalable decentralized cloud fabric. Finally, we provide a detailed overview of a platform, referred to as a hybrid edge cloud (HEC), designed and developed according to these requirements and principles.

2. FUNDAMENTALS OF CLOUD DECENTRALIZATION

Cloud decentralization has several advantages. As mentioned in the Introduction, in the current centralized cloud model, as more devices are added or more content is generated by these devices, more servers in data centers must be added to support them. Using a decentralized cloud, we can create a cloud fabric that scales with the number of smart devices. This reduces the need for additional servers and the upgrade cycle of these servers in datacenters. In effect, we increase the "cloud" capacity as the number of smart devices increases. In addition,, given that most of the data are produced on smart devices, we minimize the transport costs and latencies for applications. In this new model, much of the

10.1109/MCOM.001.2200251 © 2022 IEEE Published in
IEEE Communications Magazine September 2022

processing is performed on devices, communication is kept as local as possible, and heterogeneous smart devices from different vendors and operating systems can collaborate and share computing and other resources. The central cloud remains a valuable resource because it may be indispensable for many applications that require global management, central storage, or processing.

Data center resources need to increase, but at a reasonable pace, to accommodate the needs for central processing only and relegating all other possible tasks and functions to smart devices where most of the data are generated. Servers in data centers will no longer be the bottleneck or the "always necessary" trust elements and do not need to grow in proportion with smart devices, but only in proportion to the needs of central processing as dictated by use cases and applications.

2.1 Challenges of Cloud Decentralization

There are several challenges to building a pragmatic decentralized cloud platform. The first is fragmentation in operating systems and networks. For a decentralized cloud to become feasible, devices must connect, communicate, and collaborate across many fragmented operating systems and networks.

The second major challenge is the availability of network resources. Once smart devices act as servers, they must connect and communicate with other devices by using uplink network resources. Although network connectivity is gradually becoming symmetrical, today, there are more downlinks than uplink resources available (download speeds on most consumer and enterprise networks are better than uplink speeds).

To better describe the challenge, Figure 3 illustrates an example of posting a video from a smartphone (acting as a client) to the central cloud. We assume that the video is consumed by two smart phones and a personal computer (PC). In the centralized cloud model, the video is first uploaded to a server in the data center, and notifications are sent to recipients to download the video from the server (which acts as a storage and streaming server). In the decentralized cloud model, the video can be streamed directly from the smartphone (acting as a steaming server) to the three destination smart devices. In the centralized case, we have one instance of uplink and three instances of downlink. In the decentralized case, we have three instances of uplink (assuming none are behind a firewall). From a bandwidth efficiency point of view, the decentralized cloud approach has an advantage; when we consider the asymmetric nature of most of the networks today, uplink resources may become a challenge and impact user experience.

Thankfully, the telecom industry has become aware of this, and most new wired and wireless standards now specify more symmetric connections where uplink and downlink resources are balanced. However, it is important to be cognizant of this when building applications using decentralized clouds.

10.1109/MCOM.001.2200251 © 2022 IEEE Published in
IEEE Communications Magazine September 2022

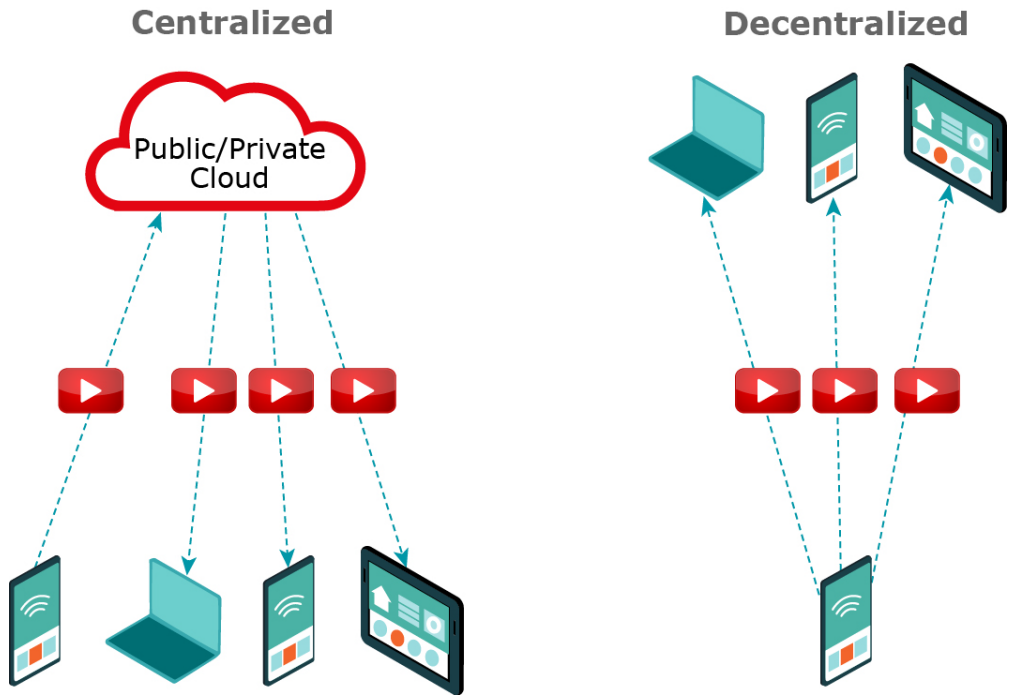

Figure 3. Video sharing example using centralized cloud vs decentralized cloud.

The third major challenge is that, unlike servers in data centers, most smart devices may be non-persistent in nature. There is less control over their availability and reliability, especially for battery-operated and mobile devices. Unlike servers in datacenters, there is little central control over these devices. For instance, in the example above, if the smartphone sending the video is turned off, the video streams to the other devices stop. This must be considered when building applications where persistent nodes are essential. Persistent nodes can always be provisioned using other collaborating edge servers (such as home or enterprise gateways); in the worst case, central cloud resources can be used when required.

The fourth major challenge is the management of distribution. In data centers, distribution management deals with resource availability based on simpler criteria such as CPU load, memory constraint, and IO. The scope of distribution management is the specific data center where the solution is running. In a decentralized cloud, the criteria for distribution management are much more diverse and include power availability and consumption, reliability, and device capabilities. As discussed later in Section 2.3.2, with decentralized cloud, distribution scopes expand to devices on the same network, within a given proximity, and user accounts because most devices belong to specific users.

The fifth challenge is security because, in a decentralized cloud, it is not feasible to have firewalls surrounding all available resources. Consequently, every device on the network should be treated as a potential rogue device, and appropriate security measures are required to ensure that rogue nodes cannot disrupt secure operations within the network of devices.

10.1109/MCOM.001.2200251 © 2022 IEEE Published in
IEEE Communications Magazine September 2022

Despite these challenges, the benefits of a decentralized cloud far outweigh its challenges, and all of these challenges can be overcome in a well-designed system, as described in this paper.

2.2 Drivers and Requirements for Cloud Decentralization

Figure 4 shows the high-level architecture of "centralized cloud" and compares it to "decentralized cloud". With a decentralized cloud, all nodes, including the severe farms in data centers, referred to as the central cloud in the figure, can act as cloud servers, and there is no designated permanent central element. Nodes can communicate, collaborate, and share resources directly, generally without resorting to a central element, unless necessary. With this approach, the central cloud resources are used only when required. For instance, when there is a need for global storage, archiving, updating of centralized databases, centralized registration, and so on. Any other functions that can be handled by smart devices at the edge can be assigned to them. For instance, messaging between devices, handshaking control signals between machines, and transmitting data between devices within a small cluster.

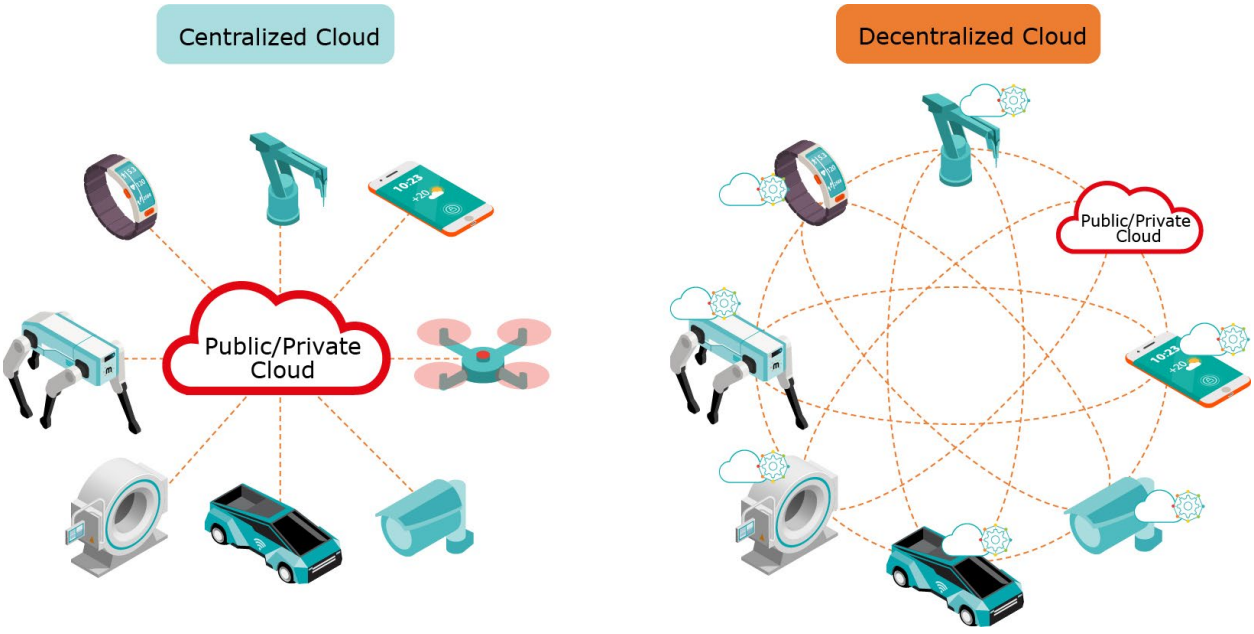

Figure 4. Centralized versus decentralized clouds.

The explosion of "client" devices on the edge of the network and the change in role from consumers of data to producers/consumers of data are major drivers of cloud decentralization. However, other trends in the software industry make cloud decentralization necessary and feasible. One important trend is the adoption of microservice architecture.

The complexity of managing software solutions made up of a large number of components in the past has led to monolithic solutions. However, with the evolution of virtualization technology toward lighter container management platforms, the consumerization of on-demand IT, the management of the collection of microservices (service mesh and cluster

10.1109/MCOM.001.2200251 © 2022 IEEE Published in
IEEE Communications Magazine September 2022

management), and the ease of rich communication (API), this complexity has drastically decreased. A good software design practice involves developing solutions as a collection of many instances of single-purpose, well-defined components called microservices [25] [26] .

The consequences of designing systems in this way are:

- more granular utilization of infrastructure resources to closely follow the demand curve
- simplification of the design of complex attributes (session, tenancy)
- better distribution and utilization of computing resources within or between data centers
- memory-efficient and power-efficient architecture that loads memory and runs only active microservices at any time
- further decomposition of solution clients from a monolithic to microservice architecture for faster application development time and ease of software upgrade and maintenance.

To achieve even more efficiency of the software solution, a current trend is to have ephemeral microservices, also referred to as server-less architecture, where microservices are instantiated (launched and run) based on API calls made to the microservice itself to achieve even more efficiency of the software solution. Programming using ephemeral microservices is referred to as a serverless architecture that leads to a more memory efficient and power efficient architecture that loads to memory and runs only active microservices at any time.

As previously discussed, a major challenge with offloading computing to smart devices is the possible lack of persistence in both the connectivity and power status of these devices. These challenges may be overcome using a microservice approach.

The cloud can be extended to include smart devices by recognizing and exposing their computing resources and utilizing them in an opportunistic manner when available. Adding analytics to the way ephemeral microservices are deployed based on availability, policy, and context (including social and other application-level events) can help to optimally deploy applications on the decentralized cloud.

To make decentralized cloud computing practical and useful, several considerations must be considered. We need to ensure that existing smart devices can act as cloud servers when needed. The developers should be able to build applications with as little effort as possible. Given the heterogeneous nature of these devices, we should be able to assign functional roles based on device capabilities. To make life easier for developers, we should follow API semantics similar to the major existing public cloud platforms. We should also provide a light container to run microservices on any smart device and, in that light, follow the semantics of existing major container technologies as much as possible.

From the perspective of smart devices, there are several essential requirements. Smart devices should have the following capabilities to become potential cloud servers.

- Discover the existence of other computing nodes of any type, regardless of the OS or the network.

10.1109/MCOM.001.2200251 © 2022 IEEE Published in
IEEE Communications Magazine September 2022

- Discover other nodes' capabilities and behavior (hardware specifications, OS, persistency, connectivity, etc.).
- Discover microservices supported by other nodes.
- Dynamically form clusters along with other nodes, especially around the network, proximity, and user accounts.
- Communicates with other nodes (including central cloud resources) at the microservice level, either directly or through other nodes across different clusters.
- Connects with other nodes if they choose to share data, services, or resources.
- Adapt to functions and roles based on their resources and capabilities.
- Process and analyze data locally when possible.
- Be as secure and trustworthy as a centralized cloud, using a zero-trust model.

We suggest a platform-agnostic approach to develop downloadable application-level software to enable any computing device to act as a cloud server when possible, and as a result, build an end-to-end decentralized cloud platform. To make this feasible for the industry and developer community, we have concluded that the following requirements are essential for creating a pragmatic decentralized cloud platform.

- Should require no change to the device hardware, OS Kernel, or drivers.
- Should work on most modern hardware (PCs, STBs, router tablets, smart phones, etc.).
- Should have a very small memory footprint.
- Should support hosting microservices and load, run and stop across devices.
- Ideally, support multi-tenancy, where multiple apps and microservices are hosted with a single instance of the platform software.
- Should have a light but scalable backend hosted on "central cloud" whether private or public.
- Should use bootstrap mechanisms for the registration of the nodes.
- Should create dynamic clusters of nodes within a given scope.
- It should support non-persistency (appearing and disappearing) of inter- and intra-cluster nodes.
- Should create effective persistency when needed by pulling collaborating decentralized and/or centralized resources dynamically.
- Should manage communication between nodes either directly or through intermediate nodes.
- Should dynamically instantiate backend resources based on demands from the nodes.

Various principles must be considered to create a pragmatic and scalable architecture to unleash the power of smart devices and create a massive decentralized cloud. This section describes several important principles.

10.1109/MCOM.001.2200251 © 2022 IEEE Published in
IEEE Communications Magazine September 2022

2.3 Basic Principles of Cloud Decentralization

In this section, we describe some of the fundamental principles that ensure that cloud decentralization is pragmatic, efficient, and secure.

2.3.1 First Principle: Meritocracy

Meritocracy is a key principle in ensuring an efficient system design in which the use of network bandwidth and central resources is minimized. All nodes should have equal opportunities to participate in the network based on the merit and value they provide. Nodes may play a role based on their capabilities and characteristics. The capabilities of any node should be maintained in the node profile. For instance, a node with large storage can be selected dynamically as a cache node or a backup storage node, a node with high network connectivity can be a proxy node, and a persistent node can become the holder of knowledge for a cluster of nodes.

Meritocracy is essential for decentralization. Otherwise, we must provide central elements with predefined roles, which may lead to an inefficient hierarchical structure. Ideally, any cluster of two or more nodes should be able to operate and collaborate based on merit.

Several characteristics are necessary to establish meritocracy. Thus, transparency is essential. All the participating nodes should tell the truth regarding their profiles in a transparent manner. However, meritocracy cannot be applied optimally. The architecture should remove the incentives to lie (not providing node-specific privileges or rights). Even when there is no apparent incentive to lie, we need a mechanism to blacklist the nodes that lie about their profile to harm the operations of a cluster.

It is also important to consider that a meritocracy may change over time. Nodes can upgrade or downgrade their capabilities and profiles. The architecture must accommodate any changes to the nodes in real time.

Perhaps the most important consideration with regard to the meritocracy principle is *the merit of the central cloud resources* in this new architecture. The centralized cloud architecture is a special case of a decentralized cloud in which smart devices are used only as clients. Therefore, it is easy to become complacent and assign most tasks back to the central cloud even when there is no merit. It is essential to avoid existing bad practices and avoid falling back on readily available third-party resources on the central cloud to speed up development while sacrificing hosting costs, latency, and privacy. For the meritocracy principle to work effectively, all nodes should be considered as potential servers to other nodes, and all requests must be kept local to the cluster where a node is active.

2.3.2 Second Principle: Decentralized Discovery

A node must discover other nodes based on its scope. Some scopes are attached to the same network, owned by the same account holder or within the proximity of one another.

The discovery process should use any combination of these or other scopes without a dedicated central node such as a presence server. For instance, if a node sits behind a

10.1109/MCOM.001.2200251 © 2022 IEEE Published in
IEEE Communications Magazine September 2022

firewall and is not reachable from the outside, it should rely on any node that is reachable to become discoverable and not on a central entity unless it is necessary.

The discovery process should not be limited to information on presence and how to connect and communicate to a node, but also to some of the important characteristics and roles and personas a node can adopt: cache node (a node with spare storage), proxy node (good connectivity to the Internet), CPU resources (node with spare CPU to run microservices), etc.

2.3.3 Third Principle: Clustering

Human and machine communication occurs mostly in clusters. Robert Dunbar, an anthropologist, suggested a cognitive limit of 150 for people with whom humans can have stable relationships [27] . He explained it as "*the number of people you would not feel embarrassed about joining uninvited for a drink if you happened to bump into them in a bar*". In other words, humans communicate within constrained clusters. Additionally, we seldom regularly communicate with everyone in our close relationship circle. In fact, daily communication may be limited to a handful of very close relationships. Therefore, it makes logical sense that the communication framework considers this when assigning roles and responsibilities to nodes within a cluster.

However, cluster-based communication characteristics are not limited to humans. The communication between the machines is similar. For most applications, communication is often limited to a few nodes in a cluster at any given time. Therefore, communication should be optimized to occur locally in the cluster as much as possible. To remove the requirement for any node to have to handshake with every other node, we need an architecture in which one node (which we call the supernode) in a cluster is given a special role as the knowledge holder of the cluster and is assigned the role of communicating that knowledge to global discovery or nodes in other clusters.

The system should allow nodes to form their own ad hoc clusters dynamically based on certain scopes, as described above. Nodes should dynamically assume roles via election or selection by other nodes based on a series of characteristics of nodes and rules. Thus, the nodes can dynamically form the fabric of a decentralized cloud (i.e., software-defined cloud infrastructure). As nodes enter and exit clusters, their roles should be reassigned dynamically based on merit.

Following the creation of clusters, nodes can discover, connect, and communicate within and across clusters directly, or via the dynamic instantiation of intermediator nodes. This bootstrap model helps to avoid overloading any nodes, whether global or local, and therefore reduces traffic and chattiness and creates a light and scalable architecture. Given the potential non-persistence of the nodes, presence notification should be left to the node itself, along with the responsibility to decide which other nodes it wants to notify. Therefore, there is no need for a single global presence server or registration point. Similarly, there should be no need for “keep alive” mechanisms at the infrastructure level between the nodes. These types of mechanisms can be delegated to microservices, if needed.

10.1109/MCOM.001.2200251 © 2022 IEEE Published in
IEEE Communications Magazine September 2022

2.3.4 Fourth Principle: Microservice to Microservice Communications

Once a decentralized cloud fabric is formed, applications on devices can utilize it to communicate directly without a pre-assigned third-party trust element. However, this is not sufficient to connect nodes at the physical network level. We must ensure direct and secure communication at the microservice level. All nodes, including smart devices on the edge, should be able to join the service mesh and communicate directly. In addition, any node should be able to load, start, and stop microservices on any other node. This ensures that microservices running across the platform can communicate without the need for a fixed central entity.

Microservices enabled on smart devices should follow the standard architecture by exposing them through their own API(s) and using fine-grained scopes to protect the resources exposed via APIs. The decentralized cloud platform should enable seamless reachability of microservices across nodes to form a service mesh either directly or via a similar pattern [28] described in more detail in Section 3.1.3. In environments that can run container daemons (e.g., Linux), a decentralized cloud platform should provide functionalities for the communication of these microservices across all OS(s), devices, and networks. In environments that cannot run container daemons (e.g., smartphones), the platform should provide additional "light" container capabilities with the ability to download, deploy, and operate microservices.

2.3.5 Fifth Principle: Dynamic Resource Instantiation

For decentralization to be efficient, there should be very little overhead associated with enabling communication across nodes that are not accessible directly. For the lack of a better term, we refer to this as a dynamic resource instantiation.

Signaling and data resources should be deployed dynamically based on the network conditions and demand from nodes within clusters. As a result, there is no need to pre-assign communication resources, but dynamically instantiate and dismantle these resources once the function is completed. This increases efficiency and reduces cost by deploying endpoints that are instantiated only when needed. The platform should assist the nodes in setting tunneling opportunistically to increase signaling and data bandwidth efficiency. Resources should be deployed based on certain parameters of network topology and demand by the application. These communication resources should be prioritized to instantiate the closest proximity of the cluster, otherwise in the private or public cloud.

2.3.6 Sixth Principle: Collaboration

To leverage their collective power, all nodes, including smart devices, must collaborate and share resources. The sharing of decentralized cloud resources should be seamless, as it is

10.1109/MCOM.001.2200251 © 2022 IEEE Published in
IEEE Communications Magazine September 2022

with server nodes in the central cloud. As a first step, we should be able to use the collective resources of all computing devices.

For instance, recording a video on a smart phone and seamlessly storing the recorded content on a personal computer, network-attached storage (NAS), or even a connected storage dongle. As a next step, all nodes should be able to share their resources with others. For instance, allowing family members to share an NAS as a family resource, or allowing colleagues to share computing and communication resources in an enterprise. Ultimately, nodes should be able to lease computing resources to others and create an even larger, decentralized cloud.

That said, we must be careful not to tightly tie the decentralized cloud to collaboration. A decentralized cloud provides the opportunity to take advantage of collaboration and resource sharing across nodes. However, even without device-level collaboration, a decentralized cloud can provide several benefits. As a fundamental step, any application built on a smart device should prioritize using its local resources to host microservices to service other nodes in its cluster based on the requirements of the application. In other words, Jack's device should be used as a server to host Jack's app. However, with collaboration, one can go further and use resources from other nodes. For instance, Jill's phone can run a microservice for Jack's application even when they are not in an active session, Jack can provide spare storage for Jill's videos on his device, or Jill can use Jack's fiber connection instead of her poor cellular connection.

In other words, collaboration can significantly improve efficiency and scaling; however, it is not necessary to make cloud decentralization impactful.

2.3.7 Seventh Principle: Infrastructure Independence

For cloud decentralization, the decentralized cloud platform must be agnostic to the operating systems, central cloud platforms, networks, and locations. Many failed industries have attempted to standardize the decentralized communication between nodes. There are many corporate and government interests involved in varying and conflicting agendas.

Intellectual property issues create a significant barrier in creating a homogeneous system. More importantly, it is best to allow operating systems and networks to evolve to provide a riper environment for innovation and fundamental disruption. Otherwise, we will face even more issues with legacy protocols, modules, libraries, and data.

Therefore, the decentralized cloud platform itself must be independent of the evolution of operating systems and networks. In other words, the platform should operate on top of the existing operating systems and networking standards at the application layer. This is the most pragmatic way to ensure that the platform is deployed and maintained over the long term.

10.1109/MCOM.001.2200251 © 2022 IEEE Published in
IEEE Communications Magazine September 2022

2.3.8 Eight Principle: Zero Trust Security

As mentioned in Section 2.1, it is not feasible to create firewalls around all devices in an edge cloud cluster. In a decentralized cloud environment, a zero-trust security model is essential for ensuring secure communication and operation.

Therefore, security needs to be ensured at many levels by encrypting the communication between devices, encrypting the payload, and protecting all resources using keys with fine-grained scopes. Thus, building a zero-trust network, encrypting all communications, and authenticating every device is feasible, given that at any time, a limited number of devices are active within every edge cloud cluster of devices.

3. INTRODUCTION TO HYBRID EDGE CLOUD (HEC)

In this section, we describe Hybrid Edge Cloud (HEC), a decentralized cloud platform designed and developed based on the above principles. The platform enables almost any computing device to act as a cloud server to run microservices when feasible and plausible in a completely decentralized fashion agnostic to hardware platforms, operation systems, and underlying networking technologies.

Figure 5, the HEC platform is an end-to-end system made up of a cloud management backend and the Edge Software Development Kit (Edge SDK). This is a decentralized and liquid architecture; therefore, every element can reside anywhere on any reachable computing device. The HEC platform and microservices running within can run on any operating system and communicate over any network anywhere in the world and are independent of the hardware infrastructure and central cloud platforms.

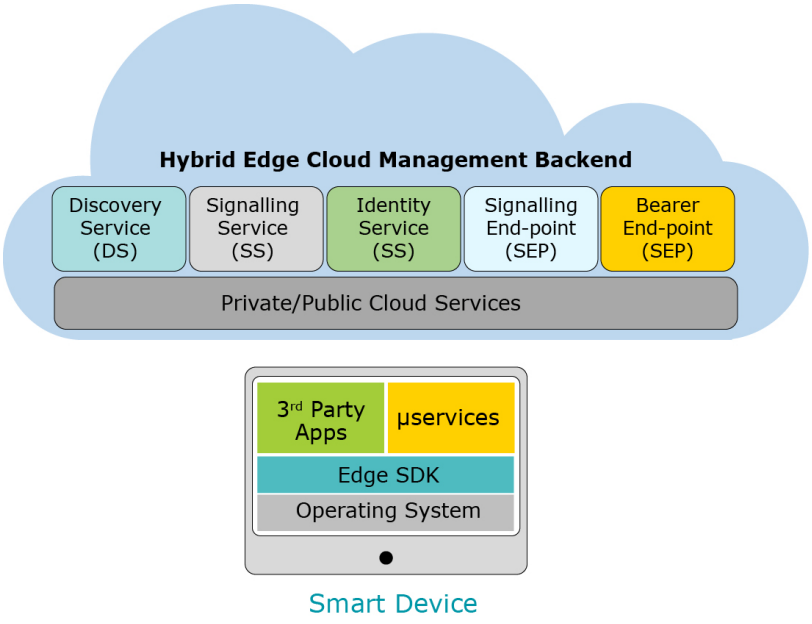

Figure 5. HEC architectural building blocks.

10.1109/MCOM.001.2200251 © 2022 IEEE Published in
IEEE Communications Magazine September 2022

The software component referred to by the Edge SDK enables any smart device to act as a cloud server. It is a collection of software libraries and their corresponding APIs. Edge SDK can run on any PC, server, mobile device, fixed gateway, autonomous car gateway, connected TV, or even in the central cloud, depending on the application use case. Once the Edge SDK is loaded, the device becomes an HEC node.

The HEC nodes:

- Can dynamically discover each other independent from the OS and/or the network.
- Can expose available capabilities and functionalities via API to each other.
- Can form and organize into clusters (edge clusters).

  Can communicate within a cluster even with no Internet availability (for the special case of link-local clusters) and across clusters.

The platform operates via the formation of clusters. Nodes in a cluster discover, connect, and communicate with other nodes. This bootstrap model is used to avoid overloading any nodes, whether global or local, and therefore, reduces traffic and chattiness and creates a light and scalable architecture. Given the potential non-persistence of the nodes, a presence notification is left to the node itself, along with the responsibility to decide which other nodes it wants to notify. Therefore, there is no need for a single global presence server or point of registration. Similarly, there is no need for "keep alive" mechanisms at the infrastructure level between the nodes. These types of mechanisms can be delegated to microservices, if needed.

As explained previously, the Edge SDK can reside on any computing device or server and is available for various hardware platforms and operating systems. It is application-level software that can be downloaded on many types of computing devices. Backend management services can be hosted on a central cloud or any reachable and reliable computing resource with sufficient computing and memory to provide the necessary services to support the edge nodes. We describe these elements in detail in the following sections.

3.1 Edge SDK Components

Figure 6 shows the major Edge SDK components. The Edge SDK resides between the operating system and the end-user application. The developers can develop their own microservices that can be hosted on the device using the Edge SDK container manager. The runtime environment for microservices is also provided by the Edge SDK.

10.1109/MCOM.001.2200251 © 2022 IEEE Published in
IEEE Communications Magazine September 2022

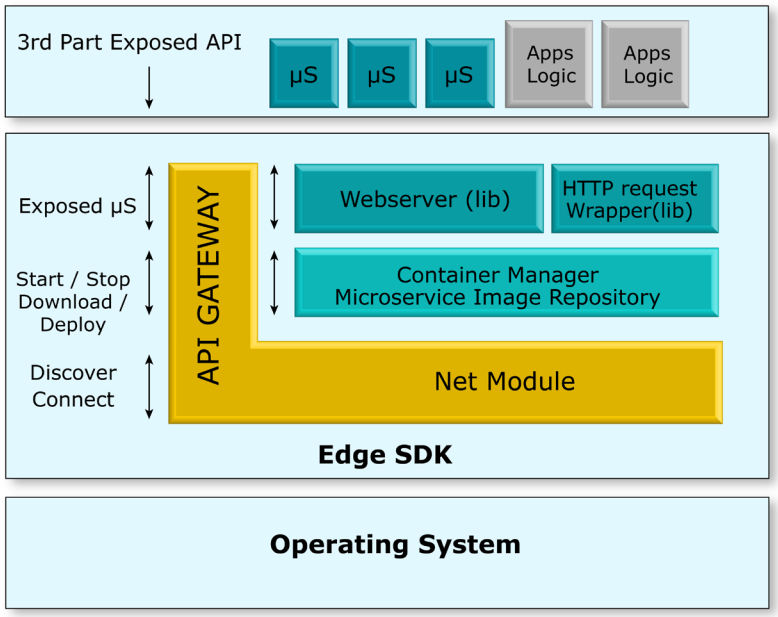

Figure 6. Edge SDK High Level Components.

By incorporating Edge SDK, computing devices are transformed into intelligent network nodes that can form clusters. Edge SDK removes the complexity of networking among nodes, enabling developers to focus on their solutions in a microservice model, even on small computing devices.

Nodes in a cluster can assume a specific role or a combination of roles, depending on the physical hardware capability, OS, attached network connectivity, types of microservices running on each node, and usage/privacy policy settings. Some roles are assigned through an election process, considering other nodes within the cluster at any given time, whereas others are assigned through a process of selection. One of the most important roles in a cluster is that of a supernode elected by all the member nodes. In the trivial case of a single-node cluster, a node serves as its supernode. A supernode is the bearer of information regarding a cluster and all its member nodes; it is the "single source of truth" for the cluster. The information maintained is related to nodes, microservices deployed on each node, and historical artifacts from Edge SDK operations. The supernode is responsible for assigning roles such as link-local proxy and link-local cache to the other nodes in the cluster.

The examples of selected nodes are:

- The link-local proxy node supports communication in cases where the cluster nodes reside behind a firewall.
- The link-local cache node can be a node with large amounts of physical storage, which can be assigned a link-local cache role for the cluster.

10.1109/MCOM.001.2200251 © 2022 IEEE Published in
IEEE Communications Magazine September 2022

For each node, the Edge SDK can support microservices and apps from different providers (otherwise called "tenants") on a device that belongs to one user. In other words, even if a user has loaded multiple apps on a smart device (such as a smart phone), all of which employ EdgeEngine, functionalities, and capabilities are related to (and authorized for) that user.

Edge SDK provides discovery, connection, and communication among devices at both physical and microservice levels:

- **Node and service discovery:** auto-discovery and auto-routing for all nodes with Edge SDK in local and global networks (s)
- **Node and service connection:** ad-hoc edge cloud of nodes forming a self-organizing cluster
- **Light container:** Microservice runtime environment to allow remote/local load of microservice images, start, stop of microservices
- **Sidecar pattern:** enabling frontend application decomposition to abstract certain functionalities (e.g., networking, load balancing, security, authentication, etc.) into a sidecar and making API calls versus dealing with libraries and complexity of cross-OS support.

3.1.1 Scopes for Clusters of Nodes with Edge SDK

Nodes with Edge SDK can discover, connect, and communicate with each other. Discovery is a "filtered search" operation based on the following three scopes:

- **Network**: nodes that are members of the same link-local network cluster. In this case, the link-local identifier is formed by combining the public IP address and– the local network address.
- **Proximity**: nodes that report themselves as physically present within a geographical location or within an area defined by a geospatial query (e.g., [30]), independent of the network to which each node belongs.
- **User account**: nodes registered under the same account ID. For this purpose, Edge SDK employs an authorization protocol through an identity SaaS provider.

The discovery process can use any combination of these scopes or new scopes can be defined. Microservices on each of these nodes and across clusters can form their own service meshes by calling each other via APIs.

Nodes and microservices running on nodes have unique identifiers: a specific microservice on a specific node is addressable uniquely, both locally and globally.

10.1109/MCOM.001.2200251 © 2022 IEEE Published in
IEEE Communications Magazine September 2022

3.1.2 Light Container: Microservice Runtime Environment

Microservices enabled on nodes expose their services through a common embedded web server. Edge SDK complements container daemons in two different ways. In environments (e.g., Linux) that can run container daemons, Edge SDK provides functionalities to manage ad hoc clusters of edge nodes, as described previously. In environments that cannot run container daemons (e.g., smart phones), Edge SDK provides additional "light" container capabilities with the ability to download, deploy and operate microservices. The embedded web server provides a subset of container management APIs.

3.1.3 Sidecar Pattern

The side-car pattern [28] allows an application to be decomposed into components built using different technologies. Using the side-car pattern, any component of an application can be built and deployed in isolation. The latency is reduced because of the proximity of the side car with the application, and components and functionality can be added without changing the application.

The sidecar pattern abstracts many complexities in dealing with the service mesh. This is possible because many of these complexities are independent of the type of microservice deployed across the edge cloud. However, the side-car pattern does not hide the decentralized nature of the network. For example, an API gateway or security token management system can be built using a sidecar pattern.

3.1.4 API Gateway

The API gateway is part of the net module within the Edge SDK and makes the API endpoints for each service accessible to all other nodes in a cluster. By providing this API gateway, the Edge SDK provides functionalities that abstract the complexity of dealing with other microservices in different clusters.

- **Security**: At the edge, security is a crucial aspect of how microservices communicate. Certain elements, such as firewalls and network partitioning, are common in the central cloud but do not generally exist on the edge. Therefore, it is necessary to address three security levels.
  - It is not possible to use https on the link-local cluster because the nodes in this cluster do not have domain names. Therefore, communication between nodes within the same link-local network is encrypted.
  - The API of each microservice is protected via fine-grained scope tokens. Generally, Edge SDK operates in a zero-trust environment. Therefore, we cannot assume that firewalls protect microservices running on edge nodes. Dealing with a valid and non-expired token is abstracted by the side car.
  - Because there are some special nodes that may manage data from other nodes (e.g., cache node or link-local proxy node), the user payload must be encrypted so

that it is only visible to authorized parties. Acquiring the key and encrypting and decrypting the user payload are also abstracted by the sidecar.

- **Routing:** For proximity and user account clusters, routing to the proper node is a complex operation that requires dealing with the supernode and link-local proxy node. The side car hides this complexity from the developer of the microservice, and they only need to invoke the appropriate microservice within the cluster.
- **Retry:** Decentralized systems require retry mechanisms to ensure fault tolerance. The sidecar handles retry calls and retries strategies. Developers can focus on developing their microservice rather than with the complexity of decentralized systems.

Similar to backend technologies, such as Istio, which helps developers handle a service mesh (a solution based on microservices talking to each other), Edge SDK handles the service mesh at the edge and deals with all the constraints of using edge devices as servers.

3.2 HEC Management Backend

HEC management backend services are hosted on servers that are reachable through the Internet and provide necessary services to support edge nodes across edge clouds. An HEC cluster is defined as a collection of nodes, each with a globally unique ID, based on the context or scope. As explained previously, a given node at any time may be a member of any or all three clusters: user account, network, or proximity clusters. Other scopes may be defined for the grouping of nodes; however, for most applications, these three scopes are sufficient.

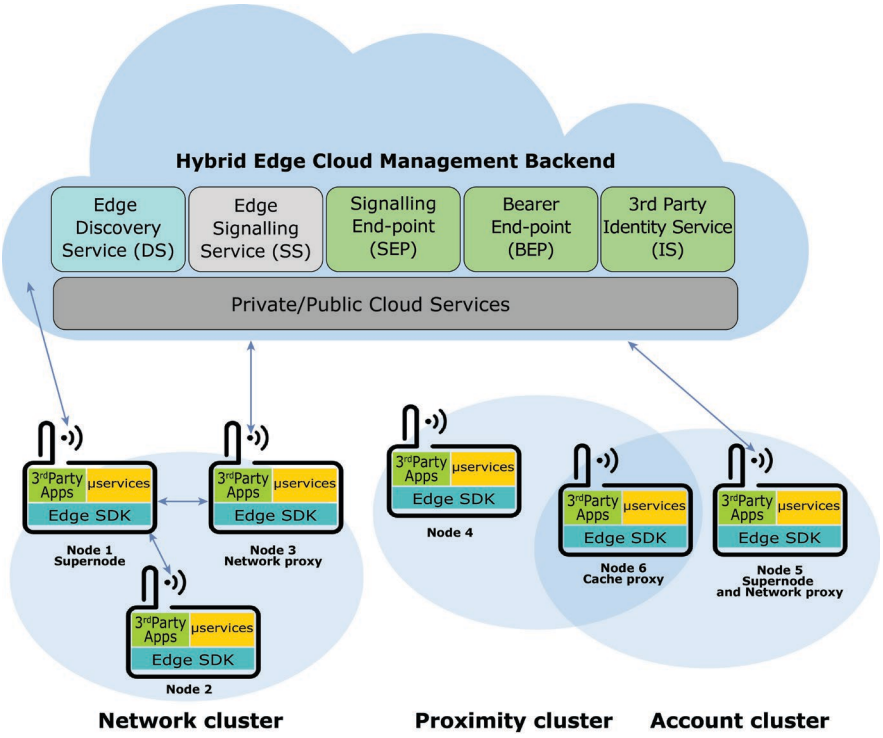

Figure 7. Components of HEC architecture.

10.1109/MCOM.001.2200251 © 2022 IEEE Published in IEEE Communications Magazine September 2022

The major elements of the HEC management backend shown in Figure 7 are:

- DS: Discovery Service
- SS: Signaling Service
- SEP: Signaling Endpoint (deployed dynamically and on demand)
- BEP: Bearer Endpoint (deployed dynamically and on demand)
- IS: Identity Service, using any third-party SaaS provider

Parts of SS and IS live both on the backend and edge nodes: link-local network proxies in each cluster are assisted by SS via SEP or BEP, and the supernode of a cluster is assisted by DS. The architecture departs from the traditional notion of "service by servers in the cloud - client on the edge". Its value comes from the distribution of services over the entire range, from the central cloud to the edge nodes, as depicted in **Figure 8.**

The SS is used to provide APIs to launch the SEP and BEP components. The SS keeps track of the existing BEP and SEP in a cluster of SS and provides the information needed to properly load balance the BEP and SEP in the cluster of SS. To provide optimal latency based on where the BEP and SEP are needed, the DS and SS are independently geo-decentralized.

The DS holds knowledge to form clusters, the overall status of the clusters, and the nodes within them. Once a cluster is formed, any new node registers with the supernode that subsequently informs DS via the supernode. To reduce traffic for scalability, updates from the supernode to the DS occur in an opportunistic fashion and only when a change occurs in the cluster.

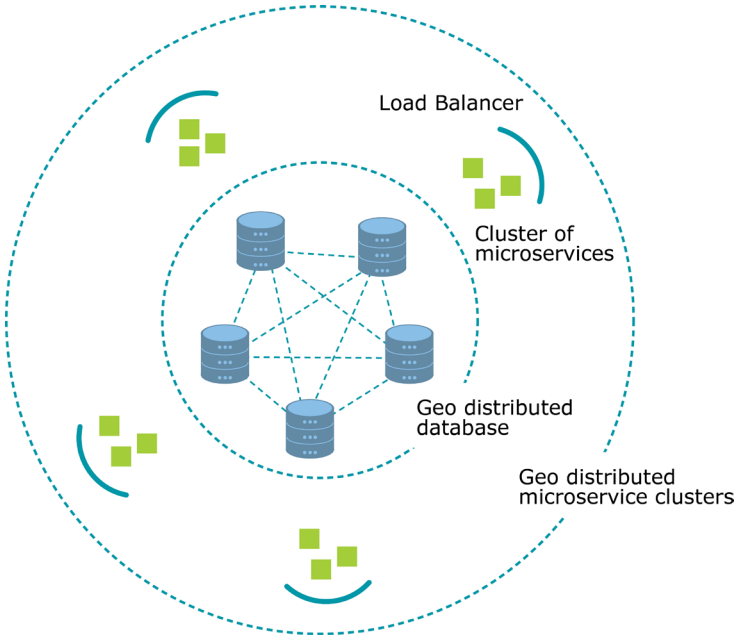

Figure 8. HEC management backend microservice distribution.

Another important function of DS is the reachability test for a supernode. When a supernode registers itself, the DS tests for reachability. The supernode might be behind a firewall, and while it could initiate a call to the DS, the DS or other external nodes might not be able to imitate a call to the supernode. In such cases, the DS requests that the SS dynamically deploy an SEP for the cluster. DS returns the SEP address to the supernode. Further descriptions of SEP and BEP are presented below.

The DS holds a complete inventory of the nodes and cluster profiles. This inventory includes details of computing resources on all nodes, status of each node, location of each node, services available on each node, end-to-end network topology to reach each node and the clusters, reachability of the clusters, availability of resources, and other pertinent information. In other words, the DS has complete visibility to all resources across the network and can supply this information to dynamically deploy services on any available resource within the network in real time.

3.2.1 Signaling Endpoint (SEP) & Bearer Endpoint (BEP)

SEP and BEP are resources that can be deployed dynamically by the SS based on the demand from the nodes within the clusters. Consequently, there is no need to reserve the computing resources. This increases efficiency and reduces cost by deploying endpoints only when needed. SEP is used for signaling communication, while BEP is used for data communications and jointly assists the nodes to setup tunneling opportunistically to increase signaling and data bandwidth efficiency.

SEP and BEP are deployed based on parameters such as time to live, number of concurrent connections, and communication protocols (HTTP, SSH, web socket, or UDP tunneling). If desired, endpoints can be deployed on the available computing resources within the closest proximity of the cluster.

The mechanics of the signaling (SEP) and bearer (BEP) endpoints are best illustrated by the example depicted in Figure 9.

In the example, it is assumed that two nodes (Node 2 in cluster networks 1 and 4 in cluster network 2) belong to the same user and have already registered with their respective link-local network clusters. The platform provides the SEP as a reachable endpoint to Node 4 to communicate with Node 2, as if it were directly accessible. After signalling is established, a BEP is provided for the bulk of the exchange between the two nodes. The flexibility of separating the signaling and bearer channels allows the creation of service-specific BEPs that are not restricted to HTTP-based service delivery.

Steps for discovery, connection and communication amongst nodes include:

- sending discovery requests to the supernode for nodes that belong to a scope
- obtaining a list of nodes together with appropriate signaling information
- sending requests to remote nodes via a SEP
- having remote nodes request a BEP for providing a service
- connecting and communicating to consume the service through the BEP provisioned

10.1109/MCOM.001.2200251 © 2022 IEEE Published in
IEEE Communications Magazine September 2022

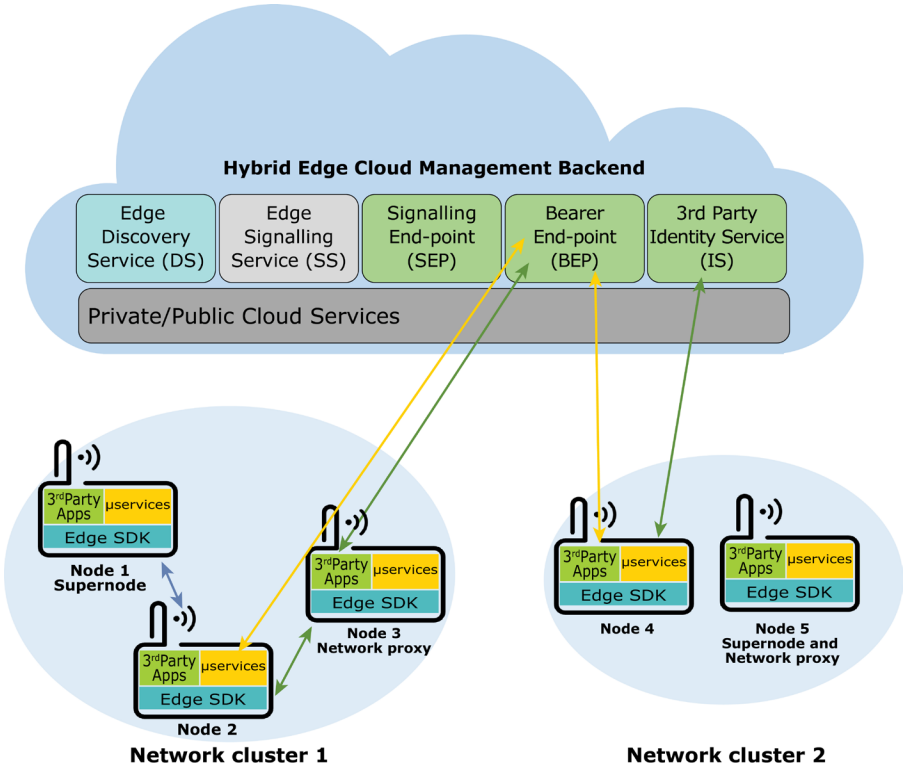

Figure 9. Discovery, connection, and communication for two devices belonging to the same UID.

3.2.2 Identity Service (ID)

Identity management can be any SaaS-based solution that resides in the central cloud, and creates and maintains the authentication profile. The platform performs authorization, which requires token generation and management. The token holder can be the Edge SDK, the microservice using the Edge SDK, the application developer using the Edge SDK, or the end-user of the application. The token is used to verify the credentials, legitimacy of the token holder, and authorizing access to all backend services using existing mechanisms such as Jason Web Tokens (JWT) [31] and a subset of standards defined in [32] to verify the identity of the token holder.

3.3 Benefits of HEC for 5G and Beyond

To make 5G efficient and economically viable, communication service providers must install computing resources on their deployment sites to minimize service latency and reduce capex and opex costs. Service providers plan to use ETSI (MEC) or GSMA (3GPP SA6) standards to deploy and operate application components on the network equipment. Moreover, these standards consider a device to be attached to the network. Utilizing the HEC concept, network-attached devices can be leveraged to run applications and

- Offload unnecessary traffic over the network by enabling policy enforcement, data processing, and caching on edge devices.
- Reduce energy consumption and the carbon footprint of the MEC/3GPP infrastructure in their base stations, given that edge devices no longer need to constantly send traffic over the network.
- Provide a more responsive and efficient solution to their enterprise customers.
- Manage higher-value data instead of raw data, which may have significant duplication/noise, and as a result, improve the value/cost ratio of their MEC/3GPP infrastructure.
- Provide cross-industry solutions to their customers and ecosystem synergy across their customer base and seamlessly connect highly fragmented MEC/3GPP silos.
- Build and maintain systems with lower Capex and Opex: Today, these systems rely on centralized application servers that are the focal points of all interactions between all devices and waste significant network and computing resources.
- Increase revenue by offering a differentiated private enterprise network as a service with higher data privacy and security.
- Provide a foundational approach for GDPR for their enterprise customers.
- Support GDPR compliance with built-in privacy and reduced complexity.
- Support environmentally friendly solutions and reduce carbon footprint.
- Avoid heavy device orchestration by relegating many functions to furthest edge of the network.
- Empower native microservice developers to build applications instead of being limited to embedded systems for specific industry specialists.
- The resiliency and robustness of their solutions are increased by minimizing the points of failure through decentralization and localization.
- Reduce the necessary upgrade cycle of computing, storage, memory, and bandwidth resources in the infrastructure.

3.4 Application Development Using HEC

A major advantage of Edge SDK is its ability to develop front-end applications on typical client devices using the microservice concept and architecture.

The transition to microservices and HEC requires development teams to work more closely because it blends different knowledge and expertise:

- **Backend developers:** Supporting billions of smart devices as clients (e.g., IoT) places a significant burden on the central cloud. On one hand, too many resources may remain idle, waiting for signals from smart devices on the edge. Additionally, fulfilling the performance demands of an application may not be feasible. For instance, deploying a backend system in the US to support a client in Europe may not meet the latency constraints for many applications. Therefore, backend developers must leverage client resources better to support these new demands. They may be forced to offload many of the functions closer to the application, even if

10.1109/MCOM.001.2200251 © 2022 IEEE Published in
IEEE Communications Magazine September 2022

it requires deploying part of the backend system in the "client" device running the application.

- **IT/DevOps:** For a long time, IT teams have been responsible for determining and managing the infrastructure where solutions are deployed. They must consider many constraints and parameters such as deployment and operation costs, scalability, and elasticity. For most applications, the scope of the cloud infrastructure is a single data center, and the main task is to address computing and networking resource constraints. To support the explosion of devices and data at the edge, the scope should be expanded to deploy IT resources at the right time and place (generally beyond the scope of a data center). New scopes, such as proximity, account, and link-local presence, need to be considered to ensure efficient deployment and operations.
- **Frontend developers:** Frontend applications are used to perform simple tasks such as inputting and sending information to the backend and/or rendering information coming from the backend. Most complex functions are generally relegated to backend servers. However, given the explosion of data generated at the edge, many new functions must be supported on smart devices, such as caching, AR, image recognition, authorization, and authentication. As a result, front-end applications are becoming larger and more complex (e.g., the Facebook app on iOS has tripled in size to over 300 Mbytes in less than two years). Therefore, there is an opportunity to transition from a monolithic front-end app design to a microservice architecture and to decompose the front-end app subsystem into microservices. The app can then seamlessly call microservices that are local to the device, along with those running on the backend (hosted on the central cloud).

One of the many consequences of a microservice-based system is the choice between multitenancy and single tenancy. One of the major benefits of a public cloud is multi-tenancy, where multiple applications can share public cloud resources and the microservices deployed on them. However, certain applications may have to deploy microservices that must remain as single tenants for a variety of reasons, such as security or data privacy. Therefore, a hybrid approach in which one can choose whether a microservice is a multi-tenant or a single tenant is a better approach.

Another important aspect is whether a microservice is a single user or multiple users. At first glance, multiuser microservices may appear to be more desirable. However, this is not always true. For instance, if a microservice is to always serve a single user within a "client device" or a pair of "client devices, " where only one acts as a client and the other acts as a server, a multi-user platform may be inefficient. Therefore, a hybrid approach, in which one can choose whether a microservice is a multi-user or a single user, is a better approach.

As the complexity of systems increases, the benefits of a hybrid approach to both of these aspects become of paramount importance. The HEC platform can provide flexibility and ease of implementing an approach to benefit backend, frontend, and DevOps with simplicity, flexibility, redeployability, and scalability of development, as described below.

- **Backend developers** can easily transition from a multi-user microservice to a single-user microservice that resides on the closest resource to the application, that is, on the same resource that the front-end application is running. In most cases, the resource exists as the application does, and the microservice only exists if the application makes a request through the API gateway. This reduces the complexity of developing multi-user microservices and brings the serverless microservice model to

10.1109/MCOM.001.2200251 © 2022 IEEE Published in
IEEE Communications Magazine September 2022

all types of edge resources, beyond the central cloud. If serverless microservices expose their RESTful APIs, they can be utilized in a cross-domain.

- **IT/DevOps** has a smaller number of microservices to manage in the central cloud, which helps reduce the complexity and operational cost. When microservices reside close to the application need (for instance on the "client" device), we achieve ultimate horizontal scalability with minimal or even no hosting cost. The complexity is also reduced because there is no need for different infrastructure knowledge because resources at the edge appear the same (albeit with different constraints) as those on the central cloud.
- **Frontend** application developers can follow backend development methodologies and decompose the complexity of the frontend application into serverless microservice and sidecar patterns, as illustrated in Figure 10. The developer can then decide where an application is active and what microservices need to run within a cluster of nodes: on the central cloud, on a local device, or another device or gateway within the cluster. As a result, the developer has more options to break down a client application, usually written as a monolithic block, into microservices and enjoy all the benefits of the microservice architecture common in backend development: scalability, flexibility, choice of technologies, isolated impact on other modules or functions, ease of deployment, etc.

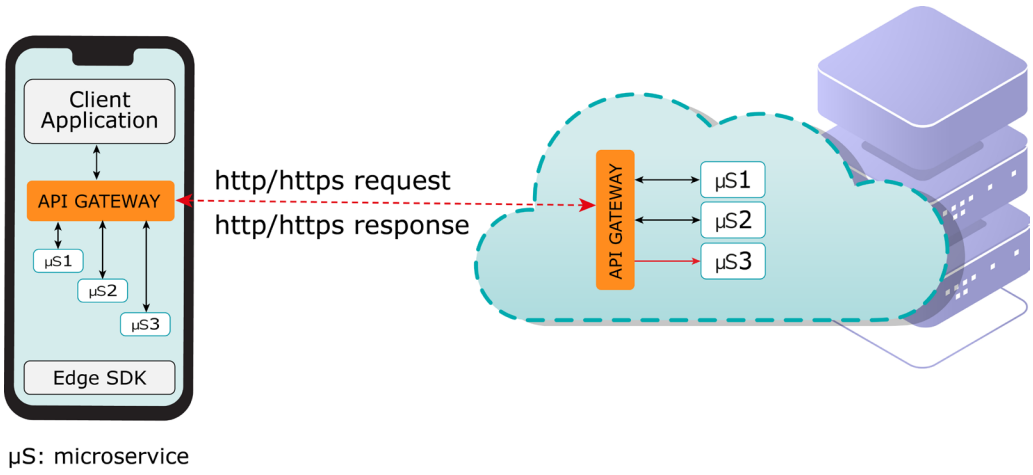

Figure 10. Client server communication using HEC with Edge SDK.

In contrast to the centralized cloud approach shown in Figure 1, the application can make requests not only to the API gateway in the central cloud, but also locally to the same device. In other words, the application can take advantage of microservices hosted locally for local functions, and globally on the central cloud for functions that cannot be hosted locally. This concept can be expanded to multiple devices and edge nodes, as shown in Figure 11 as an example of client-to-client communication.

In contrast to the centralized cloud approach shown in Figure 2, where edge devices act as clients only, client-to-client communication can occur directly between edge devices (or through servers in the central cloud), giving the developer the opportunity to optimize all aspects of deployment: cloud hosting costs, latency, bandwidth usage, data privacy, and all other benefits of the microservice architecture for typical backend functions.

10.1109/MCOM.001.2200251 © 2022 IEEE Published in
IEEE Communications Magazine September 2022

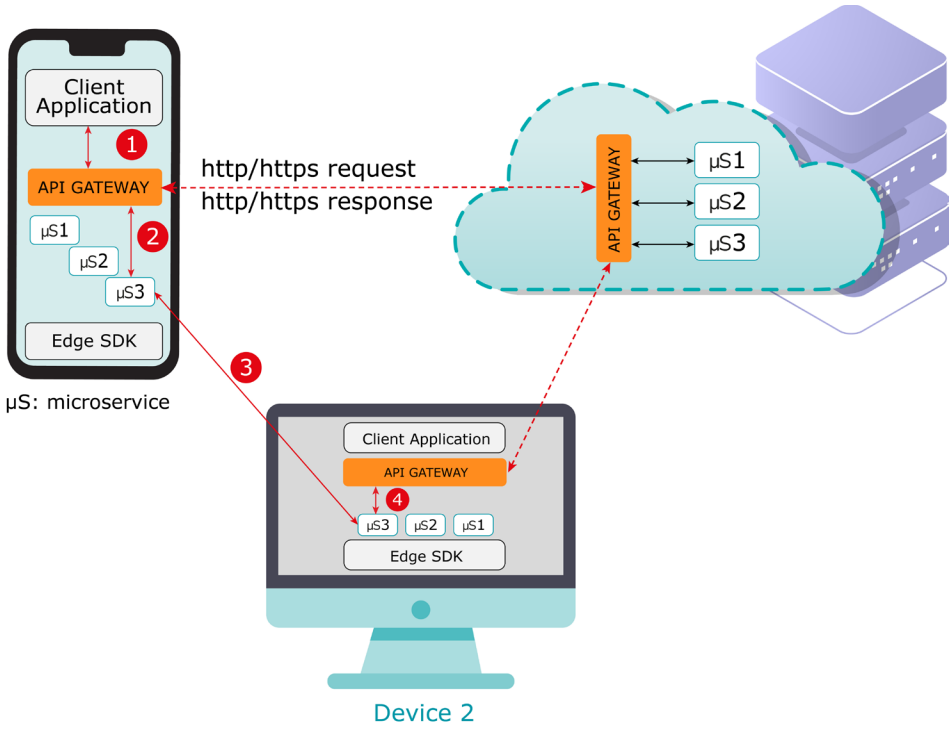

Figure 11. Client to client communication using HEC with Edge SDK.

As a result, Edge SDK benefits the developers by seamlessly expanding:

- The notion of on-demand IT resources to the edge using the same models and APIs
- The notion of clustering by adding new cluster scopes: user accounts, proximity, and networks.
- the notion of service mesh by providing a similar pattern at the edge to handle the API gateway, security, and routing for communication with other microservices, whether locally on the edge globally or in the central cloud.

In conclusion, we devised a pragmatic approach for building a decentralized cloud that can leverage the processing power, storage, and memory of billions of smart connected devices that are currently unused or seriously underutilized. This can create a cloud fabric that is orders of magnitude larger, cheaper, faster, has a lower carbon footprint, and can provide better data privacy for all consumer and enterprise applications. This will address many of the challenges of 5G deployment and help build a more efficient, equitable, and sustainable hyperconnected world.

10.1109/MCOM.001.2200251 © 2022 IEEE Published in
IEEE Communications Magazine September 2022

4. REFERENCES

[1] "5G: Personal mobile internet beyond what cellular did to telephony", IEEE Communications Magazine, Volume: 52, Issue: 2, February 19, 2014

[2] Rob van der Meulen, "What Edge Computing Means for Infrastructure and Operations Leaders", Gartner Technology Insights, October 03, 2018

[3] D. Evans, "The Internet of Things How the Next Evolution of the Internet is Changing Everything". Accessed: Dec. 3, 2016. [Online]. Available: https://www.researchgate.net/publication/30612290

[4] V. Turner, J. F. Gantz, and D. Reinsel. "The digital universe of opportunities: Rich Data and the Increasing Value of the Internet of Things". (Nov. 26, 2018). [Online]. Available: https://www.emc.com/leadership/ digitaluniverse/2014iview/index.htm

[5] "What is Cloud Computing?". Amazon Web Services. 2013-03-19. Retrieved 2013-03-20

[6] Lian Ping Chen, " Microservices: Architecting for Continuous Delivery and DevOps", The Proceeding of the IEEE International Conference on Software Architecture, ICSA 201

[7] "What's a service mesh?". Buoyant. 2017-04-25. Retrieved 5 December 2018.

[8] Adaptive Control of Extreme-scale Stream Processing Systems Proceedings of the 26th IEEE International Conference on Decentralized Computing Systems.

[9] *Nordrum, Amy (18 August 2016).* "Popular Internet of Things Forecast of 50 Billion Devices by 2020 Is Outdated"*. IEEE.*

[10] "Over 6 Decades of Continued Transistor Shrinkage, Innovation" (Press release). Santa Clara, California

[11] Characteristics of YouTube network traffic at a campus network – Measurements, models, and implications Michael Zink, Kyoung won Suh, Yu Gu, Jim Kurose

[12] https://www.statista.com/statistics/195140/new-user-generated-content-uploaded-by-users-per-minute/

[13] OECD. Participative Web and User-Created Content: Web 2.0, Wikis and Social Networking Edition complete. OCDE Information Sciences and Technologies (October 2007). http://www.oecd.org/document/40/0,3343,en_2649_34223_39428648_1_1_1_1,00.html

[14] HAMDAQA, Mohammad (2012). Cloud Computing Uncovered: A Research Landscape (PDF). Elsevier Press. pp. 41–85. ISBN 0-12-396535-7.

[15] "Should Companies Do Most of Their Computing in the Cloud? (Part 1) - Schneier on Security". www.schneier.com. Retrieved 2016-02-28.

[16] "Disadvantages of Cloud Computing (Part 1) - Limited control and flexibility". www.cloudacademy.com. Retrieved 2016-11-03.

[17] Gellman, Barton; Poitras, Laura (June 6, 2013). "US Intelligence Mining Data from Nine U.S. Internet Companies in Broad Secret Program". The Washington Post. Retrieved June 15, 2013.

[18] Greenwald, Glenn; MacAskill, Ewen (June 6, 2013). "NSA Taps into Internet Giants' Systems to Mine User Data, Secret Files Reveal – Top-Secret Prism Program Claims Direct Access to Servers of Firms Including Google, Apple, and Facebook – Companies Deny Any Knowledge of Program in Operation Since 2007 – Obama Orders US to Draw Up Overseas Target List for Cyber-Attacks". The Guardian. Retrieved June 15, 2013.

10.1109/MCOM.001.2200251 © 2022 IEEE Published in
IEEE Communications Magazine September 2022

[19] *Braun, Stephen; Flaherty, Anne; Gillum, Jack; Apuzzo, Matt (June 15, 2013).* "Secret to PRISM Program: Even Bigger Data Seizures". *Associated Press*. Retrieved June 18, 2013.

[20] *Chappell, Bill (June 6, 2013).* "NSA Reportedly Mines Servers of US Internet Firms for Data". *The Two-Way (blog of* NPR*)*. Retrieved June 15, 2013.

[21] *Staf (June 8, 2013).* "PRISM: Here's How the NSA Wiretapped the Internet". ZDNet. Retrieved June 15, 2013.

[22] *Barton Gellman & Ashkan Soltani (30 October 2013).* "NSA infiltrates links to Yahoo, Google data centers worldwide, Snowden documents say". *The Washington Post*. Retrieved October 31, 2013.

[23] PlayStation 4 sales pass over 82 million, Venture Beat, July 31, 2018

[24] AWS Global Infrastructure

[25] Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., Mitchell, N., Muthusamy, V., Rabbah, R., Slominski, A., Suter, P.: Serverless Computing: Current Trends and Open Problems. arXiv:1706.03178, June 2017

[26] Glikson, A., Nastic, S., Dustdar, S.: Deviceless edge computing: extending serverless computing to the edge of the network. In: 10th ACM International Systems and Storage Conference (SYSTOR), Haifa, Israel, May 2017

[27] Dunbar, R. I. M. (1992). "Neocortex size as a constraint on group size in primates". Journal of Human Evolution. 22 (6): 469–493. doi:10.1016/0047-2484(92)90081-J.

[28] Brendan Burns, "Designing Decentralized Systems: Patterns and Paradigms for Scalable, Reliable Services", First Edition, Feb. 2018

[29] The **serverless** trilemma: function composition for **serverless computing** Baldini, P Cheng, SJ Fink, N Mitchell… - Proceedings of the …, 2017 - dl.acm.org

[30] MongoDB geospatial query operators https://docs.mongodb.com/manual/reference/operator/query-geospatial/

[31] JSON Web Tokens RFC 7519 https://jwt.io/

[32] OpenID Connect Core 1.0 http://openid.net/specs/openid-connect-core-1_0.html