mimik

Application of mimik Decentralized Edge Cloud Platform for Manufacturing

Document Name:	Application of mimik Decentralized Edge Cloud Platform for Manufacturing
Document Number:	Version 0.1
Date:	March 16, 2018
Authors:	Siavash Alamouti

Abstract

Computing and communications technologies promise to bring more efficiency in the manufacturing process through automation and seamless data exchange. There are many current initiatives and keywords such as Industry 4.0 or Smart Manufacturing. Most of these initiatives aim to bring further efficiencies into the entire manufacturing process though digitization of the physical world and seamlessly connecting people, machines, IT systems, and even end-products in real-time. There are various disciplines involved in fulfilling the promise including Intent of Things (IoT), Cyber Physical Systems, Cloud Computing, Big Data, and Artificial Intelligence.

In this paper, we discuss how mimik's decentralized cloud platform can help modernize manufacturing in a pragmatic manner today without the need for new technology standards or massive information and operation technology (IT and OT) transformation.

1 Introduction

Smart manufacturing promises to further improve efficiencies in the manufacturing process; spanning the entire process of ideation to manufacturing and customer support and feedback. The impact to the industry is significant. Most analysts anticipate hundreds of billions of dollars of investments in modernization and trillions of dollars of benefits because of efficiencies in the next five years alone. These benefits come from significantly reduced time to market and maintenance costs, and increase in productivity and up-time of machines. There are also new opportunities for cost-effective product customization and personalization. The common vision for achieving these efficiencies is to cost-effectively connect all the elements involved in the manufacturing process including machines, IT systems, operations technologies (OT), products and even customers using the latest technology tool sets.

These tools include IoT technologies for connectivity, cyber-physical systems where physical and software components are highly integrated, big data where data from various points of the process are ingested and analyzed to improve the process along the chain, and artificial intelligence to help machines make independent smart decisions based on that data. In this paper, we describe how mimik's decentralized edge cloud platform can help realize smart manufacturing in a pragmatic manner today.

Today's central cloud architecture has been optimized for communications where people with mobile devices communicate small amounts of data real-time (voice and messaging) or consume "cloud-hosted" premium content (music videos, movies, Facebook posts, etc.). Recently, there has been an increase in content production by mobile phone users. Many people take photos and videos on their devices. The sensors inside these devices also generate data on their location, walking habits, biometrics, etc. For example, today close to 500 Million photos are uploaded on Facebook and Instagram and roughly 500 thousand hours of video is uploaded to YouTube daily; this is more than what the three major US media networks generate in content over two years!!! This is not only a consumer phenomenon but also applies to enterprises. For instance, more than 80% of businesses have started to leverage user-generated content in their marketing efforts. Today, most of this data is either consumed locally on the device or is shared with others. Users produce content on their mobile phones (edge nodes). They then upload the content to a third party "trusted" service through an app like Facebook, Instagram, or YouTube which then gets stored on the servers dedicated to that service (central cloud)[1].

The industrial and manufacturing use cases are similar in nature. Machines generate significant data through their sensors. The sensors attached through the machines are usually connected through an IoT Gateway. The IoT Gateway then connects to the cloud (private or public cloud depending on the maturity of the application or other environmental and legal constraints). Generally, machines do not communicate directly with each other or with people or processes. Almost all communication is carried through the "central Cloud".

[1] Many applications build some functions and intelligence into their native applications running on the edge device such as caching to improve the user experience. However, most applications store and process the content in the "central cloud".

Today, most machines generate relatively small amounts of data. However, future applications for IoT will encompass use cases where machines generate large amounts of data. Surveillance systems today top data generation with 20-40 MBytes/sec but there are upcoming use cases where significant amount of data is produced at the edge. For instance, self-driving cars will generate around a GByte/sec or terabytes per hour of data.

The current central cloud architecture cannot scale to meet the requirements for smart manufacturing. For instance, if data has to be shared between a handful of machines; for instance, a cluster of machines on a manufacturing line, it is more efficient to stream the data directly from the device producing the data to the ones consuming it rather than sending the data through an intermediary in the central cloud. This is especially true for rich and real-time data; for instance, video analysis of machines, people, and processes intended for real-time decisions. In other words, although most data within the manufacturing process is generated at the edge, its processing, storage, and consumption is still delegated to the “central cloud” creating inefficiencies in processing and transmission. Moreover, apart from efficiency, the necessity to upload information to the “central cloud” has created issues around privacy and security compliance. Even simple local sharing of data between machines and people on the same WiFi network often requires uploading of information to the cloud resulting in large latency, reduced bandwidth efficiency, higher cost for storage, maintenance, and protection of the data.

It is not feasible for billions of machines to connect to the central cloud potentially transmitting GBytes/sec of information per machine to be processed centrally. Instead, we should analyze the data at the edge and send only filtered or abstracted data back to the central cloud. Also, all local communication in a manufacturing plant or the field; for instance, communication with drivers’ or factory workers’ mobile phones, should happen locally to reduce latency and save bandwidth. Communication between nearby nodes should be managed locally and even communication between remote nodes should be performed directly and without a central element if possible.

Distributed edge cloud computing can potentially improve the performance (speed, latency, etc.), efficiencies (bandwidth, computing resources, etc.) of managing data distribution while improving privacy and control over the distribution of data.

In the next section, we describe the details of the mimik decentralized cloud architecture and how it can help with smart manufacturing.

2 Manufacturing & Decentralized Edge Cloud

In this section, we illustrate how decentralized edge cloud can help fulfill the vision for future smart manufacturing. One industry initiative called Industry 4.0 has identified four major widely accepted principles necessary to fulfill this vision [1]:

- Interoperability
- Information transparency
- Technical assistance
- Decentralized decisions

We will demonstrate that a decentralized cloud infrastructure where every node is turned into a cloud server is the most pragmatic way to ensure interoperability and can help with information transparency, technical assistance and making decentralized decisions.

Interoperability is the ability of all the participants in the manufacturing process to connect and communicate in real-time. This includes people, machines, sensors, and processes. Some of the challenges facing the industry in this area are incompatibility of machines in operating systems and networks, difficulty in adding consistent connectivity especially with legacy machines and processes, existing silos of Information technology (IT) and Operations Technology (OT) which are mostly hard to interface and interoperate.

The mimik decentralized cloud platform provides a simple way of ensuring interoperability between machines regardless of operating system and network technology, between people using common human interfaces such as PCs, smart phones, and tablets, and between machines and people.

This is illustrated in Figure 1. Today, the communications from sensors to the rest of the system (machines, people and processes) is generally managed through IoT gateways and the communications between the machines and other nodes is managed through central cloud. A more scalable and decentralized architecture allows for all nodes including the central cloud to connect and communicate directly.

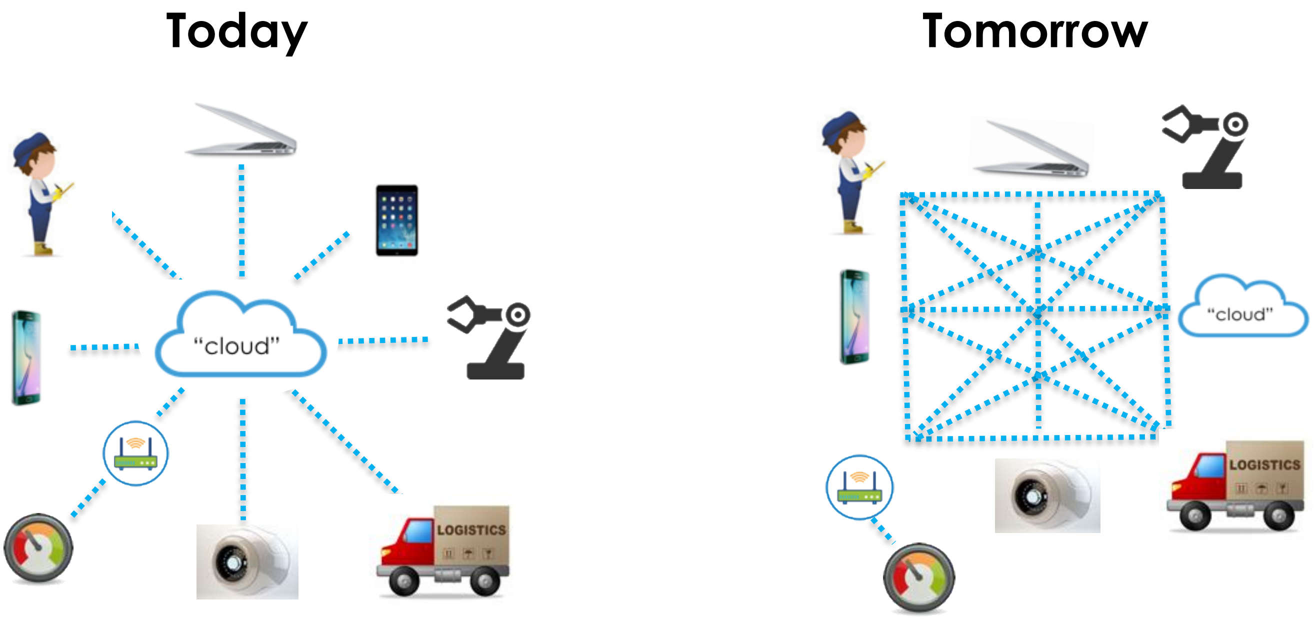


Figure 1. The evolution of the Cloud in Manufacturing.

To fulfill the architecture in Figure 1, all the nodes need to be able to communicate directly. Unfortunately, these nodes are generally incompatible (non-interoperable) today. From the perspective of an edge node, there are a list of requirements for the edge nodes to ensure interoperability. The edge nodes should have the following capabilities:

- discover the existence of other nodes regardless of OS or network
- discover other nodes' capabilities and behavior (hardware specs, OS, persistency, etc.)
- discover microservices supported by other nodes
- dynamically form clusters along with other nodes
- communicate with other nodes at the microservice level either directly or through other nodes across different clusters
- connect with other nodes if they chose to share data, services, or resources
- adapt to functions and roles based on their resources and capabilities
- process and analyze data locally
- be as secure and trustable as the central cloud.

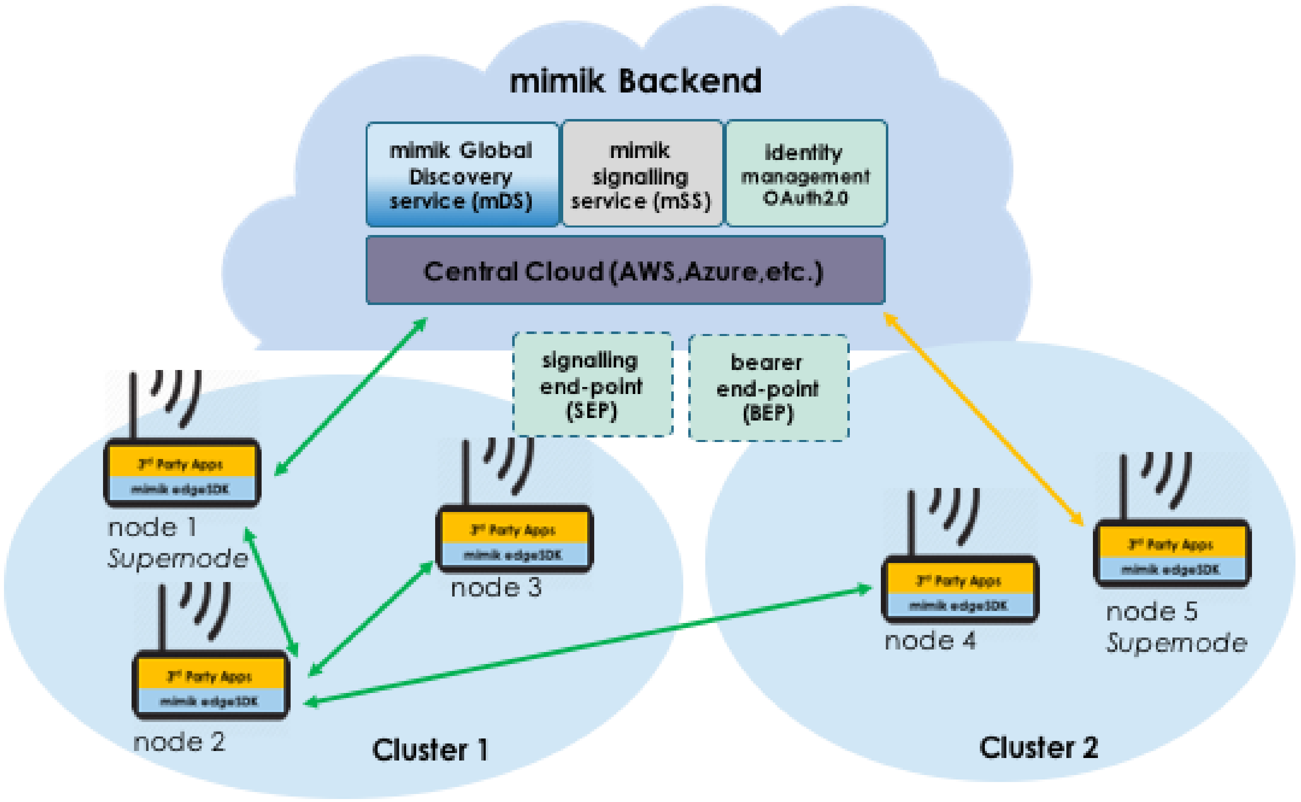
One way to meet these requirements is to create new standards and start building new capabilities at the low levels of the protocol stack for the various operating systems. However, this will undoubtedly take years and may result to bloated and ineffective standards like DLNA. Instead, we suggest a platform-agnostic approach and set ourselves the following objectives to develop downloadable application-level software that turns any computing device into an edge cloud server and as a result builds an end-to-end decentralized edge cloud platform:

- requires no changes to the device hardware, OS Kernel, or drivers
- works on most modern hardware (smart phones, PCs, tablets, STBs, etc.)
- has a small memory footprint
- supports microservices that can be easily loaded, run and stopped across devices
- supports multi-tenancy with a single instance of software to support multiple customers
- supports multiple applications and microservices with a single instance of the software
- has a light but highly scalable backend hosted on public "central cloud"
- uses bootstrap mechanism for the registration of the nodes
- creates dynamic clusters of nodes within the same network cluster, proximity and account
- manages mobility characteristics (appear and disappear) of the nodes inter and intra clusters
- manages communication between the nodes either directly or through intermediate nodes
- dynamically instantiates back-end resources based on demands from the nodes.

With these requirements in mind, we have created a platform to ensure interoperability between the edge nodes. We have created a downloadable application, with a small memory footprint for almost all operating systems. All that is needed is to install mimik edgeSDK on any machine with a computing device, gateway, or even cloud servers and create a fully connected decentralized cloud infrastructure where all the nodes can discover, connect and communicate at the microservice level. In the next section, we describe mimik's end-to-end decentralized cloud platform.

3 mimik Decentralized Edge Cloud Platform Overview

As shown in Figure 2, The mimik distributed edge cloud platform is an end-to-end system that is made up of central and edge elements: the mimik edge-Software Development Kit (edgeSDK), mimik backend services, and several microservices. This is a distributed and liquid architecture so almost every element can reside anywhere on any available computing device.



mimik Backend
mimik Global Discovery service (mDS)
mimik signalling service (mSS)
identity management OAuth2.0
Central Cloud (AWS,Azure,etc.)
signalling end-point (SEP)
bearer end-point (BEP)
3rd Party Apps
mimik edgeSDK
node 1
Supernode
node 2
node 3
Cluster 1
node 4
node 5
Supernode
Cluster 2

Figure 2. mimik architectural building blocks.

The mimik edgeSDK turns any edge device into a cloud server and extends the cloud computing infrastructure to the edge. Edge devices can be any device with basic computing capability such as machines, robots, IoT gateways, servers, smart phones, game consoles, connected TVs, car infotainment systems, etc. We hereby refer to edge devices with mimik software as nodes. These nodes have the following characteristics. They:

- dynamically discover each other independent from the OS and network,

- expose the computing and available capability and functionality to each other,
- form and organize into clusters (edge clusters),
- communicate within the cluster even with no Internet availability, and across clusters.

The platform operates by the formation of mimik cluster nodes. A mimik cluster is formed by the first active node based on a given scope (described in detail later in the document). When a node is activated (enabled with mimik edgeSDK), it first looks for a supernode which oversees global discovery and holds the knowledge of the edge cloud. If no supernode is found, the first node declares itself as the supernode. If internet is available, the supernode then informs global discovery of its existence and receives the list of nodes within the defined scope. To maintain efficiency, the supernode informs other nodes within its scope.

Following the creation of a cluster by the supernode, subsequent nodes entering the cluster, discover the existing supernode, register themselves to the supernode, and receive the list of nodes within their scope. The new nodes inform other nodes within their scope of their existence. 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 mobility of the nodes, 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 single global presence server or a point of registration. Similarly, there are no “keep alive” mechanisms at the infrastructure level between the nodes. Those kinds of mechanism are delegated to microservices if needed.

As explained before, the mimik edgeSDK can reside on any computing device or server. The edgeSDK is available for various hardware platforms and operating systems as described later in the document. It is an application-level software and can therefore be downloaded on many types of computing devices. The mimik backend services may be hosted on the central cloud for instance AWS and Azure, or alternatively on any always-connected and reachable server to provide necessary services to support the edge nodes.

We describe all these elements in detail in the following sections.

3.1 mimik edgeSDK

The mimik edge-Software Development Kit (edgeSDK) is a collection of mimik software libraries and corresponding APIs. Developers can use it to turn most computing devices into edge cloud servers regardless of OS, manufacturer, and connected network. edgeSDK can run on any mobile device, fixed gateway, autonomous car gateway, connected TV or even in the “central cloud”, depending on the application use case. Once edgeSDK is downloaded onto a device, it becomes a cloud edge node. Hereby, we will refer to any device with mimik edgeSDK as a “node”.

As shown in Figure 3, mimik edgeSDK resides between the operating system and the end-user application. There are several microservices (uS) available from mimik and 3rd parties can develop their own microservices on top of edgeSDK. A runtime environment for microservices is also provided by mimik edgeSDK. Figure 3 highlights the edgeSDK components.

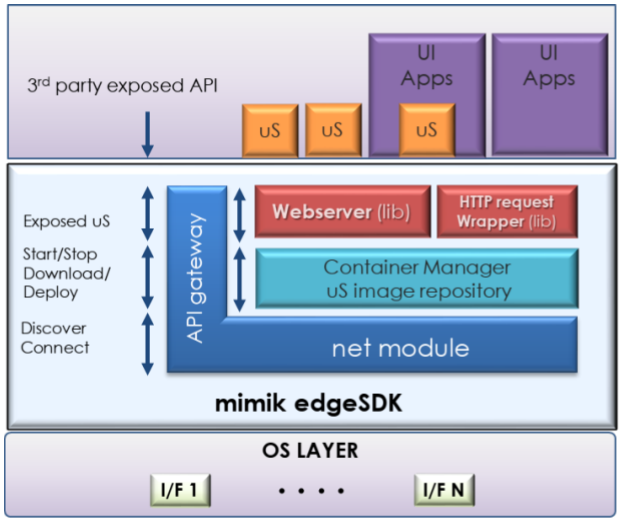


Figure 3. mimik edgeSDK high level components.

By incorporating edgeSDK, computing devices are transformed into intelligent network nodes, able to form clusters. mimik edgeSDK takes away complexity of networking among distributed edge cloud nodes, enabling developers to focus on their solution in a microservice model even on small mobile devices.

Nodes in a cluster can take a specific role or combinations of roles, depending on physical hardware capability, OS, attached network connectivity, types of microservices running on each node and usage/privacy policy settings. Some roles are assigned through a process of election, considering other nodes within the cluster at any given time, while others are assigned through a process of selection. One of the most important roles in a cluster is that of the supernode, to which a node is elected by all member nodes. In the trivial case of a single-node cluster, a node serves as its own 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. Information maintained is related to nodes, microservices deployed on each, as well as historical artifacts from edgeSDK operation. The supernode is responsible for assigning roles such as link-local proxy and link-local cache to other nodes in the cluster.

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

For each node, edgeSDK supports a unique user and multiple microservices and application providers (otherwise called "tenants"). In other words, even if a user has loaded multiple applications on a mobile device all of which employ edgeSDK, functionality and capabilities are related to (and authorized for) that user.

3.1.1 edgeSDK components

The mimik edge SDK provides Discovery, Connection, and Communication among devices, both on physical and microservice levels:

- Node and Service discovery: auto-discovery and auto-routing for all nodes with mimik edgeSDK in local and global network(s)
- Node and Service connection: ad-hoc edge-cloud of nodes form a self-organizing cluster
- Microservice management: load, run and manage microservice instances
- Edge web server: microservices runtime environment

3.1.1.1 Discover, Connect and Communicate

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

- **user account**: nodes registered under the same account ID. For this purpose, the mimik edgeSDK currently supports OAuth 2.0 based OpenID standard through a third-party Identity SaaS provider;
- **network**: nodes that are members of the same link-local cluster network. The link-local identifier in this case is formed by combining the public IP address and the link-local network address;
- **proximity**: nodes which are reporting themselves as physically present at a geographical location or within an area defined by a geospatial query (e.g. [1]).

The discovery process can use any combination of these scopes or new scopes can be defined.

Nodes and microservices running on nodes have unique identifiers: a specific microservice (e.g. mimik drive) on a specific node is addressable uniquely, locally and globally.

3.1.1.2 Microservice Runtime Environment

Microservices enabled on mimik edge nodes expose their services through a common embedded web-server. API end-points for each service are accessible from all other nodes in a mimik edge cluster through the API gateway which is part of the mimik net module, as illustrated in Figure 3. mimik edge SDK complements container daemons (ala Docker) in two different ways. In environments that can run container daemons (e.g. Linux), edgeSDK provides functionalities to manage ad-hoc clusters of edge nodes as described before. In environments that cannot run container daemons (e.g. smart phones), edgeSDK provides additional "light" container capabilities with the ability to download, deploy and operate microservices. The embedded webserver provides a subset of container management (e.g. Docker) APIs with the following constraints:

- must use specific language based on the underlying OS (java for android, objective c for iOS, etc.)
- microservices that run on the edgeSDK "light" container environment must use the web server provided by mimik edgeSDK to optimize the usage of limited resources on the underlying platform.

3.2 mimik Backend Services

The mimik backend services may be hosted on the cloud; for instance, AWS and Azure, or on other always connected and reachable servers, and provide necessary services to support the edge nodes across the edge clouds. An edge cloud is defined as a collection of nodes (devices), each having a globally unique ID, based on context or scope. As explained before, a given node may be a member of three clusters: user account cluster, which is the cluster of nodes belonging to the user that registered them (and all other accounts tied to it); the network cluster is the link-local network cluster to which the nodes in the cluster are physically connected; or the proximity cluster which is the cluster of nodes within a certain surrounding area.

The major elements of the mimik cloud backend are:

- mDS: mimik Global Discovery Service
- mSS: mimik Signaling Service
- SEP: Signaling End-Point (deployed dynamically and on demand)
- BEP: Bearer End-Point (deployed dynamically and on demand)
- IS: Identity Service, using third party SaaS provider

Parts of mSS and mDS live both on the backend and on edge nodes: network proxies in each cluster are parts of mSS; supernodes are part of mDS. The mimik architecture departs from the traditional notion of "service in the cloud - client on the edge"; its value comes from distribution of services over the entire range, from central cloud all the way to the edge nodes, as depicted in Figure 4.

3.2.1 mimik Global Discovery Service (mDS) and mimik Signaling Service (mSS)

The mDS holds the 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 mDS via the supernode. In the interest of reducing traffic for scalability, updates from the supernode to the mDS happen in an opportunistic fashion and only when a change occurs in the cluster.

The other most important function of the mDS is the reachability test to a supernode. When a supernode register itself, the mDS test for reachability. The supernode might be behind a firewall and while it can initiate a call to mDS, the mDS or other external nodes might not be able to imitate a call to the supernode. In such cases, the mDS will request the mimik Signaling Service (mSS) to dynamically deploy a Signaling End-point (SEP) for the cluster. The mDS returns the SEP address to the supernode. Further descriptions of SEP and BEP are below.

The mDS holds a complete inventory of nodes and cluster profiles. This inventory includes: details of computing resources on all the nodes, status of each node, location of each node, services available on each node, the end-to-end network topology to reach each node and the clusters, the reachability of the clusters, the availability of resources and other pertinent information. In other words, mDS 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 real-time. To make things easier for developers, standard amazon semantics are used for exposing the resources in a similar fashion as central cloud resources.

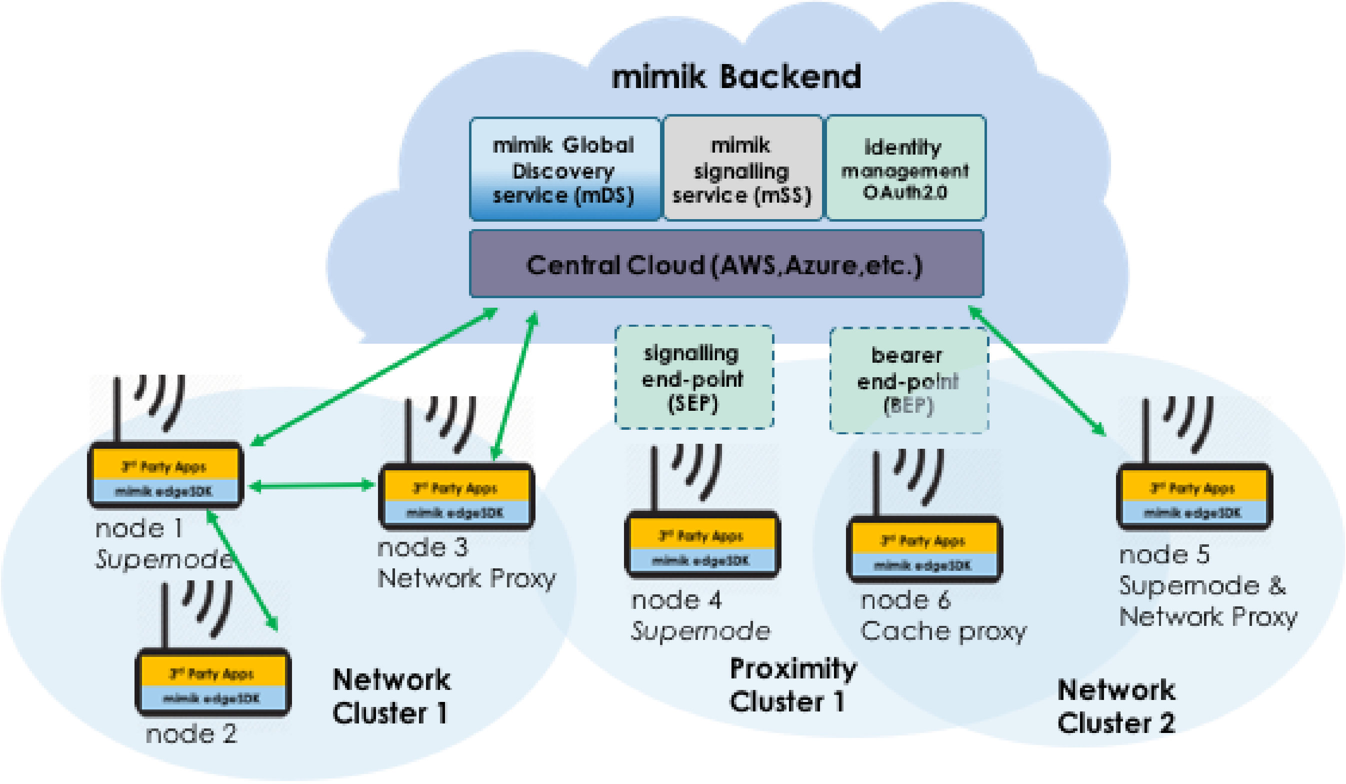


Figure 4. Components of the mimik Edge architecture

3.2.2 Signaling End-Point (SEP) and Bearer End-Point (BEP)

The SEP and BEP are resources that can be deployed dynamically by the mSS, dynamically based on the demand based on nodes within clusters. As a result, there is no need to reserve computing resources. This increases efficiency and reduces the cost by deploying the end points only when needed. The SEP is used for signaling communication while BEP is used for data communications and jointly they assist 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, end points can be deployed on an available computing resources within the closest proximity of the cluster.

The mechanics of the signaling (SEP) and bearer (BEP) end-points can be illustrated best via the example depicted in Figure 5. It is assumed that two nodes (node 2 in cluster network 1 and node 4 in cluster network 2) belong to the same user and have already registered with their respective link-local network clusters. The mimik edge architecture provides the SEP as a reachable end-point for node 4, that it can use to communicate with node 2 as if it were directly accessible. After signaling is established, a BEP is provided for the bulk of the exchange among two nodes. The flexibility of separating 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

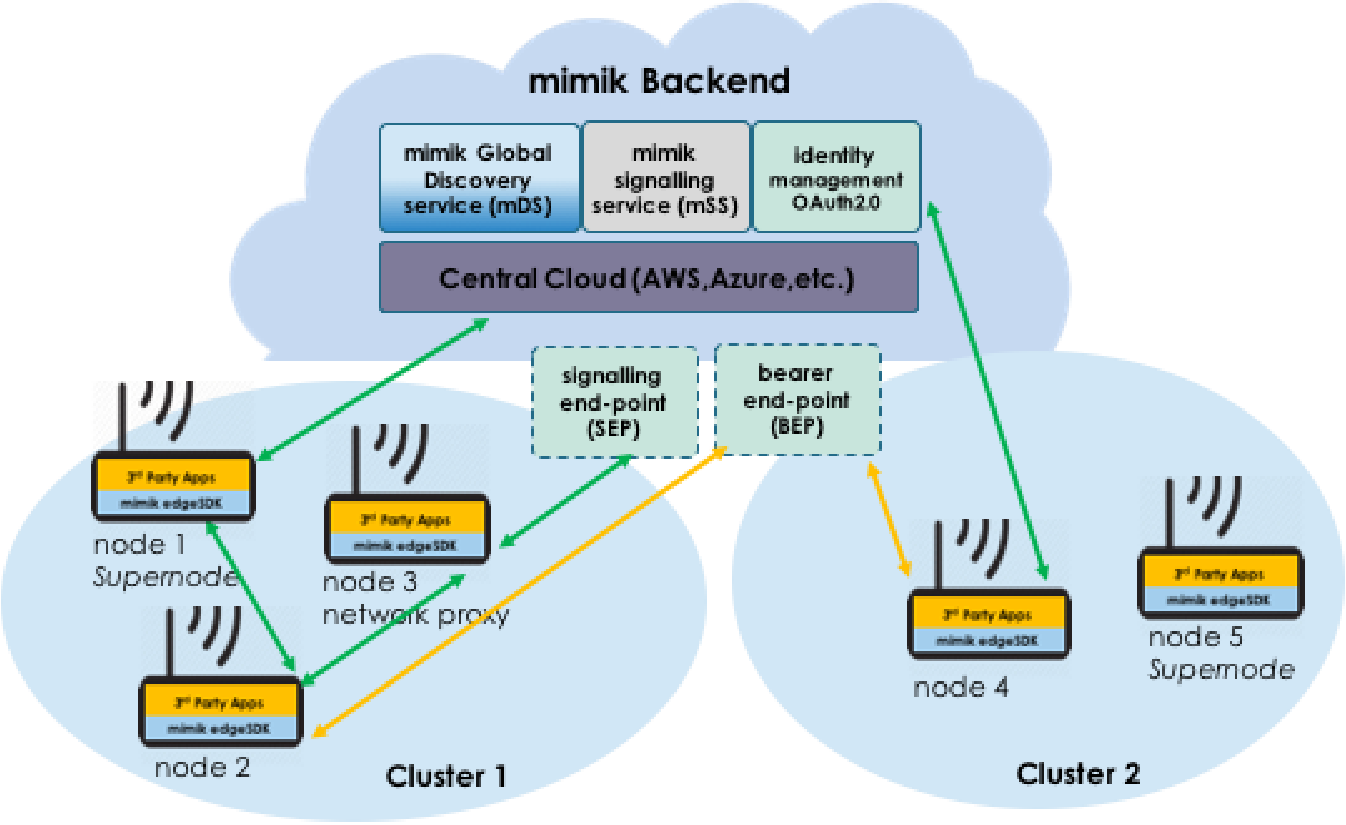

mimik Backend
mimik Global Discovery service (mDS)
mimik signalling service (mSS)
Identity management OAuth2.0
Central Cloud (AWS,Azure,etc.)
signalling end-point (SEP)
bearer end-point (BEP)
3rd Party Apps
mimik edgeSDK
node 1
Supernode
node 3
network proxy
node 2
Cluster 1
node 4
node 5
Supernode
Cluster 2

Figure 5. Discovery, connection and communication for two devices belonging to the same UID.

3.2.3 Identity Service (IS)

The mimik edgeSDK employs third-party identity software as a service (SaaS), currently based on the OAuth2.0, which resides in the public cloud and creates and maintains the authentication profile. The mimik platform manages authorization (token generation and management) for the token holder. The token holder can be the edgSDK, the microservice using the edgeSDK, the application developer using the edgeSDK as well as the end-user of the application. The token is to verify the credentials, legitimacy of the token holder, and authorizing access to all mimik backend services using Jason Web Tokens (JWT) [2] and a subset of standard “claims” defined in [3] for verifying the identity of the token holder.

3.3 mimik Microservices

mimik offers various microservices, utilizing the edgeSDK, to speedup application development and enable developers to immediately take advantage of distributed edge. Here are some examples”

- mimik drive microservice: abstracts access to storage available on edge nodes and provides distributed file management via a popular API
- mimik beam microservice: beams content from a node to node(s) and/or to service(s), peer-to-peer, one-to-one and one-to-many
- mimik transcoding microservice: provides hardware-accelerated real-time dynamic transcoding of video streams on edge nodes.

4 Supported platforms

Today, edgeSDK is available for the following platforms:

- Linux kernel 3.0 and above
- Raspberry Pi Raspbian 8.1 and above
- Windows 10
- OS X 10.11 (El Capitan) and above
- Android 4.4.2 and above (GMS or AOSP)
- iOS version 9.0 and above

4.1 System requirements

4.1.1 Minimum Operating System Version:

- Linux kernel 3.0 and above
- Raspberry Pi Raspbian 8.1 and above
- Android 4.4.2 and above (GMS or AOSP)
- iOS version 9.0 and above
- Windows 10
- OS X 10.11 (El Capitan) and above

4.1.2 Minimum Hardware Requirements:

- CPU: 2 cores ARM CPU or more (or equivalent on CISC or MIPS processors)
- RAM: recommended 256 MBytes

4.1.3 Minimum Compiler Requirement:

- C++ version 11 on all supported OS platforms

5 Integrating the mimik edgeSDK

The mimik edgeSDK provides native class wrappers (or API wrappers) for all supported platforms, in the interest of shortening the learning curve and accelerating development. These are available for:

- Android, developed in Java
- iOS, developed in Objective-C
- Linux, Windows or Mac OS X, developed in C++

6 Bibliography

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

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

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

7 References

[1] Hermann, Penta, Otto, 2016: Design Principles for Industry 4.0 Scenarios, accessed on 4 May 2016