TECHNICAL REPORT

ISO/IEC TR 23188

First edition
2020-02

Information technology — Cloud computing — Edge computing landscape

Reference number
ISO/IEC TR 23188:2020(E)

Contents

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see http://patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 38, *Cloud computing and distributed platforms*.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

Edge computing is increasingly used in systems that deal with aspects of the physical world. Edge computing involves the placement of processing and storage near or at the places where those systems interact with the physical world, which is where the "edge" exists. One of the trends in this space is the development of increasingly capable Internet of Things (IoT) devices (sensors and actuators), which generate more data or new types of data. There is significant benefit from moving the processing and storing of this data close to the place where the data is generated.

Cloud computing is commonly used in systems that are based on edge computing approaches. This can include the connection of both devices and edge computing nodes to centralized cloud services. However, it is the case that the locations in which cloud computing is performed are increasingly distributed in nature. The cloud services are being implemented in locations that are nearer to the edge in order to support use cases that demand reduced latency or avoiding the need to transmit large volumes of data over networks with limited bandwidth.

This document aims to describe edge computing and the significant elements which contribute to the successful implementation of edge computing systems, with an emphasis on the use of cloud computing and cloud computing technologies in the context of edge computing, including the virtualization of compute, storage and networking resources.

It is useful to read this document in conjunction with ISO/IEC TR 30164[1)] [27], which takes a view of edge computing from the point of view of IoT systems and the IoT devices which interact with the physical world.

1) Under development. Current stage 10.99.

Information technology — Cloud computing — Edge computing landscape

1 Scope

This document examines the concept of edge computing, its relationship to cloud computing and IoT, and the technologies that are key to the implementation of edge computing. This document explores the following topics with respect to edge computing:

— concept of edge computing systems;

— architectural foundation of edge computing;

— edge computing terminology;

— software classifications in edge computing, e.g. firmware, services, applications;

— supporting technologies, e.g. containers, serverless computing, microservices;

— networking for edge systems, including virtual networks;

— data, e.g. data flow, data storage, data processing;

— management, of software, of data and of networks, resources, quality of service;

— virtual placement of software and data, and metadata;

— security and privacy;

— real time;

— mobile edge computing, mobile devices.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 22123-1:—[2], *Information technology — Cloud computing — Part 1: Terminology*

ISO/IEC TS 23167, *Information technology — Cloud computing — Common technologies and techniques*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 22123-1, ISO/IEC TS 23167 and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at http://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

2) To be published.

3.1 Edge computing

3.1.1
distributed computing
model of computing in which a set of *nodes* (3.1.5) coordinates its activities by means of digital messages passed between the *nodes* (3.1.5)

3.1.2
edge
boundary between pertinent digital and *physical entities* (3.2.8), delineated by networked *sensors* (3.2.9) and *actuators* (3.2.1)

Note 1 to entry: Pertinent digital entities means that the digital entities which need to be considered can vary depending on the system under consideration and the context in which those entities are used. See 5.2 for more detail.

3.1.3
edge computing
distributed computing (3.1.1) in which processing and storage takes place at or near the *edge* (3.1.2), where the nearness is defined by the system's requirements

3.1.4
lightweight node
node (3.1.5) with limited processing, storage and networking capacities

3.1.5
node
networked machine with processing and storage capabilities

3.1.6
edge computing system
system providing functionalities of *edge computing* (3.1.3)

3.1.7
endpoint
combination of a binding and a network address

[SOURCE: ISO/TR 24097-3:2019, 3.4]

3.2 IoT terms

3.2.1
actuator
IoT device (3.2.4) that changes one or more properties of a *physical entity* (3.2.8) in response to a valid input

[SOURCE: ISO/IEC 20924:2018, 3.2.2]

3.2.2
Internet of Things
IoT
infrastructure of interconnected entities, people, systems and information resources together with services which processes and reacts to information from the physical world and virtual world

[SOURCE: ISO/IEC 20924:2018, 3.2.1]

3.2.3
Internet Protocol
IP
protocol specified in RFC 791 (IP version 4) or in RFC 2460 (IP version 6)

[SOURCE: ISO/IEC TR 21890:2001, 3.4]

3.2.4
IoT device
entity of an *IoT system* (3.2.6) that interacts and communicates with the physical world through sensing or actuating

Note 1 to entry: An *IoT device* (3.2.4) can be a *sensor* (3.2.9) or an *actuator* (3.2.1).

[SOURCE: ISO/IEC 20924:2018, 3.2.4]

3.2.5
IoT gateway
entity of an *IoT system* (3.2.6) that connects one or more proximity networks and the *IoT devices* (3.2.4) on those networks to each other and to one or more access networks

[SOURCE: ISO/IEC 20924:2018, 3.2.6]

3.2.6
IoT system
system providing functionalities of *Internet of Things* (3.2.2)

Note 1 to entry: IoT system is inclusive of *IoT devices* (3.2.4), *IoT gateways* (3.2.5), *sensors* (3.2.9), and *actuators* (3.2.1).

[SOURCE: ISO/IEC 20924:2018, 3.2.7]

3.2.7
operational technology
OT
hardware and software that detects or causes a change through the direct monitoring and/or control of physical devices and systems, processes and events in the organization

3.2.8
physical entity
entity that has material existence in the physical world

[SOURCE: ISO/IEC 20924:2018, 3.1.26, modified — Note 1 to entry has been removed.]

3.2.9
sensor
IoT device (3.2.4) that measures one or more properties of one or more *physical entities* (3.2.8) and outputs digital data that can be transmitted over a network

[SOURCE: ISO/IEC 20924:2018, 3.2.9]

3.3 Real time

3.3.1
real time
processing of data by a computer in connection with another process outside the computer according to time requirements imposed by the outside process

[SOURCE: ISO/IEC 2382:2015, 2122900, modified: words 'pertaining to' removed to improve substitutability of definition; Notes 1 to 3 to entry have been removed.]

3.3.2
real time system
system in which processing meets *real time* (3.3.1) requirements

3.3.3
hard real time system
real time system (3.3.2) whose operation is incorrect if results are not produced according to specified timing requirements

3.3.4
soft real time system
real time system (3.3.2) whose operation is degraded if results are not produced according to specified timing requirements

4 Symbols and abbreviated terms

AC	Alternating current
ASIC	Application-Specific Integrated Circuit
BYOD	Bring Your Own Device
CDN	Content Distribution Network
CSC	Cloud service customer
CSP	Cloud service provider
DDoS	Distributed Denial of Service
EPG	Electronic Programme Guide
EPROM	Erasable Programmable Read Only Memory
FPGA	Field Programmable Gate Array
Gb	Gigabyte
GPS	Global Positioning System
GPU	Graphics Processing Unit
IETF	Internet Engineering Task Force
IoT	Internet of Things
IP	Internet Protocol
IPTV	Internet Protocol television
LAN	Local Area Network
MDM	Mobile Device Management
OS	Operating system
PC	Personal Computer
PII	Personally Identifiable Information

RAM	Random-access Memory
RFC	Request for Comments
ROM	Read Only Memory
TPM	Trusted Platform Module
VM	Virtual Machine
VoIP	Voice over Internet Protocol
VPN	Virtual Private Network

5 Overview of edge computing

5.1 General

Over time, the forms of computing have varied between centralized and distributed, depending on the nature and capabilities of the computing devices and of the networks used to connect them.

Mainframe computers represent a form of centralised computing, where the main computer systems are placed in a data centre, containing processing and storage units. Originally, almost the whole of the computing system was situated within the data centre. Gradually, time-sharing terminals were located in remote locations to provide user access to the mainframe systems. Terminals were typically little more than a display with a keyboard for input and the associated network connection had limited bandwidth, perhaps involving a dial-up modem.

The personal computer (PC) represents a distributed form of computing. The PC has significant processing and storage capabilities and can be used very effectively in a standalone mode. However, PCs are more typically used in a networked mode. Initially, the networks were used for simple communications such as (text based) email, but as the network bandwidth increased over time, increasingly sophisticated activities took place, with file transfer and eventually peer-to-peer capabilities being used.

The availability of higher bandwidth networking encouraged the development of the client-server application architecture, with the PC used for the client, connected to a centralized server which performs the main processing and storage of the application. Client applications can include quite substantial software elements performing significant processing activities. Data might also be stored locally for faster access, although the main database(s) are held centrally.

The advent of the internet and the World Wide Web (WWW) represents the appearance of another form of computing. In this form of computing, web servers serve up web pages and related material which are accessed through client web browsers. Devices running web browsers can be relatively low in compute power, while the web servers for some of the more popular and high demand web sites can involve massive compute power spread over many machines in a large data centre.

Cloud computing is a computing paradigm that makes available all types of computing resources in an on-demand, highly scalable fashion via cloud services. Cloud computing in practice is made possible through a highly centralised architecture, with computing resources concentrated in large data centres. However, cloud computing in practice also has some distributed computing features. It is typical for cloud service providers (CSPs) to have multiple physically separated data centres and cloud service customers (CSCs) commonly distribute their applications and data across multiple data centres – for resilience, to reduce latency and for disaster recovery purposes. In addition, the favoured design paradigm for cloud native applications is to distribute multiple instances of each application component across different machines within the cloud computing system. This design paradigm and the technologies that support it are of significance to edge computing.

5.2 Concepts of edge computing

Edge computing is distributed computing in which data processing and storage takes place on nodes which are near to the edge. The edge is marked by the boundary between pertinent digital and physical entities, i.e. between the digital system and the physical world, delineated by networked sensors and actuators.

Pertinent digital entities means that the digital entities which need to be considered can vary depending on the system under consideration and the context in which those entities are used.

An example of varying pertinence are the servers within a cloud computing data centre. From the perspective of CSCs building systems using cloud services running on these servers, these IoT devices are anything but "at the edge". However, from the perspective of the CSPs having to manage the cloud computing data centre, it is highly likely that the servers are instrumented with a variety of sensors capable of reporting various physical properties of the servers, for example their temperature. Those sensors are at the edge.

The concept of nearness to the edge also needs explanation. Nearness for edge computing is usually based on minimising the latency for communication between the IoT devices that are at the edge and the place(s) where data processing and storage occurs. Nearness can mean placing the edge computing nodes physically close to the IoT devices. In the most extreme cases, nearness means combining the sensors and actuators and edge computing into a single node, as might happen with a smart phone. In other cases, the edge computing nodes are separated from the IoT devices but are placed physically close to the IoT devices and have a proximity network connecting them designed to minimise latency. Nearness can also be influenced by the nature of the networks and the volume of data flowing to and from the IoT devices – where large volumes of data and high data rates are involved, edge nodes are placed so as to reduce the latency of handling this data to the minimum necessary to meet the requirements of the use case.

Digital systems can observe and affect the physical world. Sensors and actuators are at the edge between the digital systems and the physical world. Edge computing systems generally combine these IoT devices with distributed computing resources to provide the capabilities of the system. In edge computing systems, actions often need to occur within specific timeframes, i.e. edge computing systems can also be real time systems, and latency considerations affect system design and the choice of the placement of data processing and storage to achieve timing requirements. Edge computing helps to meet those timing requirements.

Edge computing is characterized by networked systems in which significant data processing and storage takes place on nodes near the edge, rather than in some centralized location. Edge computing can be contrasted with centralized computing where the centralized nodes are remote from the edge. However, it is important to note that edge computing is complementary to centralized forms of computing and that in any given system, edge computing is often used in conjunction with centralized computing.

There are multiple reasons for the rise in the use of edge computing. One reason is the arrival of new devices combining significant processing power and storage with low power usage. Smart phones have been one of the driving factors in this area, with billions of such devices in daily use. The Internet of Things (IoT) is another reason, with small, low power, low cost IoT devices enabling the creation of IT systems which can sense and act on real world entities.

5.3 Architectural foundations of edge computing

Edge computing involves nodes that are highly heterogeneous and which are commonly arranged in tiers of compute and storage capabilities. A simplified view of the organization of edge computing nodes and the networks connecting them in edge computing is shown in Figure 1.

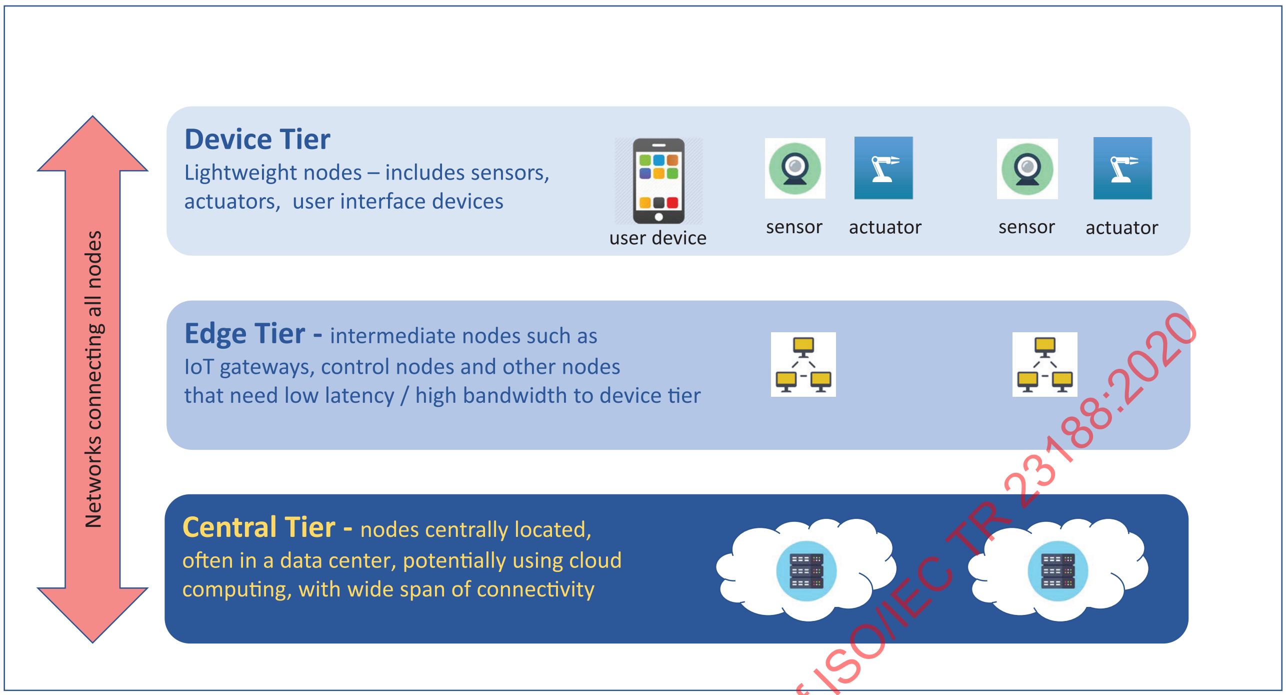
Device Tier
Lightweight nodes – includes sensors, actuators, user interface devices
user device
sensor
actuator
sensor
actuator
Edge Tier - intermediate nodes such as IoT gateways, control nodes and other nodes that need low latency / high bandwidth to device tier
Central Tier - nodes centrally located, often in a data center, potentially using cloud computing, with wide span of connectivity
Networks connecting all nodes

Figure 1 — Organization of nodes in edge computing

The tiers shown in Figure 1 are essentially a conceptual model (containing physical elements) and are illustrative rather than definitive – in reality the number of tiers and the type of node in each tier and the networks connecting them are variable, depending on the nature of the system involved. What is important to understand is that there are multiple tiers, containing varying types of nodes, all connected by networks which can also vary in nature depending on the tiers involved.

The device tier is at the edge. It typically contains lightweight nodes which commonly contain sensors or actuators or user interface devices. Such devices often have limited compute and storage capabilities. The networks used by this tier are often proximity networks, with limited bandwidth and limited range (see 6.1.1 in this document and ISO/IEC 30141:2018[2], 10.2.3.2 and 10.4.1.2 for more detail about proximity networks).

The edge tier typically sits near to the device tier (where "near" is a relative term and depends on the particular system and use case) and its role is to provide direct support to the nodes in the device tier. One type of node in the edge tier is the gateway node, for which an IoT gateway is one example. The role of the gateway node is to interconnect proximity networks to IP-based wide area networks. This may involve message and protocol syntax and semantic conversions. This role could include message encryption, deduplication and backup functions.

Another type of node in the edge tier is the control node. The control node receives data from nodes in the device tier – typically data from sensors or input from user interface devices – and responds by issuing instructions to other nodes in the device tier. Other types of node may be placed in the edge tier to meet other edge computing functional requirements. This may include management nodes, security nodes and software support nodes.

Control nodes are usually placed in the edge tier due to issues of latency and timing. The response of a control node is often time constrained (sometimes called real time – see Clause 14), such that the response must be given before some deadline following the receipt of some data or an event. One factor in this time constraint is the transmission time of messages to and from nodes in the device tier – this leads to the need for the nodes in the edge tier to be placed physically close to the device tier nodes and to the need to reduce the number of hops that the messages must take. These constraints can also influence the type of proximity network used and the protocol used over those networks. Similarly, the control nodes must have appropriate processing capacity and storage for the processing that is necessary to produce appropriate and timely responses.

As an example, if the input data is a video stream from a camera device, which is a type of sensor, and the processing required is an analysis of the video to detect the movement of some object with the intent of influencing the movement via some actuators (which are different devices from the camera), this could take a substantial amount of processing power and also require the handling of a substantial amount of data – the control node must have appropriate processing power and storage to successfully carry out its task.

The central tier represents a tier of nodes provided by centralized facilities. The nodes in the central tier offer the ability to provide very substantial compute power and storage. The central tier is an excellent place to conduct analytics or other processing that requires both a lot of compute power and access to a lot of information. The central tier can hold large stores of information which can come from many sources – this can be from across the other tiers or from outside locations, potentially sourced from other organizations. The central tier can provide services to the other tiers, including services for processing data in various ways or for holding information or providing information as required. The central tier usually has a wide span of connectivity, meaning that it is commonly connected to many other systems, including many of the distributed nodes in both the edge tier and in the device tier.

It is often the case that the central tier is implemented using cloud computing. A fuller description of the relationship of edge computing to cloud computing is given in 5.4.

It is typical of the nodes in the central tier to communicate to the other tiers using high bandwidth networks, typically the internet but possibly dedicated networks. It is also possible for the nodes in the central tier to be arranged in a highly available resilient configuration, with multiple instances of applications and services allied to replicated or redundant copies of information.

The tiers described in Figure 1 can become blurred when considering the many different types of device that are available. A significant example is the smartphone, which combines a number of elements into a single device, as follows:

— a number of sensors of various types, including GPS (location sensor), accelerometer, barometer, health monitors (such as heart rate monitoring);

— camera – both for static images and video;

— microphone & loudspeaker for audio;

— display screen and user interface;

— significant compute power (e.g. quad or octo core systems, 2 to 4 Gb RAM);

— significant local storage (e.g. up to 256 Gb);

— networking and connectivity.

These capabilities in a single device span the device tier and the edge tier and provide for dynamic addition and update of software on the device, enabling a very wide range of capabilities. Combined with excellent networking capabilities, smartphones enable some forms of edge computing in their own right – with the added advantage of their being mobile.

5.4 The relationship of edge computing to cloud computing

Edge computing can exist on its own, without any relationship to cloud computing. In terms of the tiers described in 5.3, systems can exist in which cloud computing is not used in any of the tiers. This implies that the system has no need of the capabilities offered by cloud computing. Older industrial systems are of this nature – designed to be self-contained and with fixed functionality.

However, for many systems cloud computing or cloud computing technologies are used in one or more of the tiers. This is especially so for the central tier – it is very common for the nodes in the central tier to be part of a cloud computing environment, either a public cloud or a private cloud. Cloud computing can also be used in the edge tier – as more powerful and lower-cost hardware becomes available, it is increasingly possible and desirable to use cloud computing in the edge tier. Although today use of cloud computing in the device tier is unusual and rare, there is nothing in principle to preclude its use once the device nodes are sufficiently powerful and well-connected.

It is worth noting that edge computing rarely exists on its own, but is connected to both processing and information which is held in the central tier. This means systems have processing and storage spread right across the various tiers and types of nodes. The principle is the right placement of processing and storage elements. Right placement in that processing and storage take place on nodes that are best suited to the task involved.

Figure 2 illustrates how both IoT and edge computing can relate to different parts of the cloud computing ecosystem. This can include cloud services built on a private cloud (both on-premises and more remotely) and a public cloud (including public clouds designed to serve a specific jurisdiction, or multinational or even global cloud services). Hybrid clouds of various kinds can also be employed according to business needs, as in the example shown in Figure 2 where a public cloud in the central tier is combined with a private cloud in the edge tier.

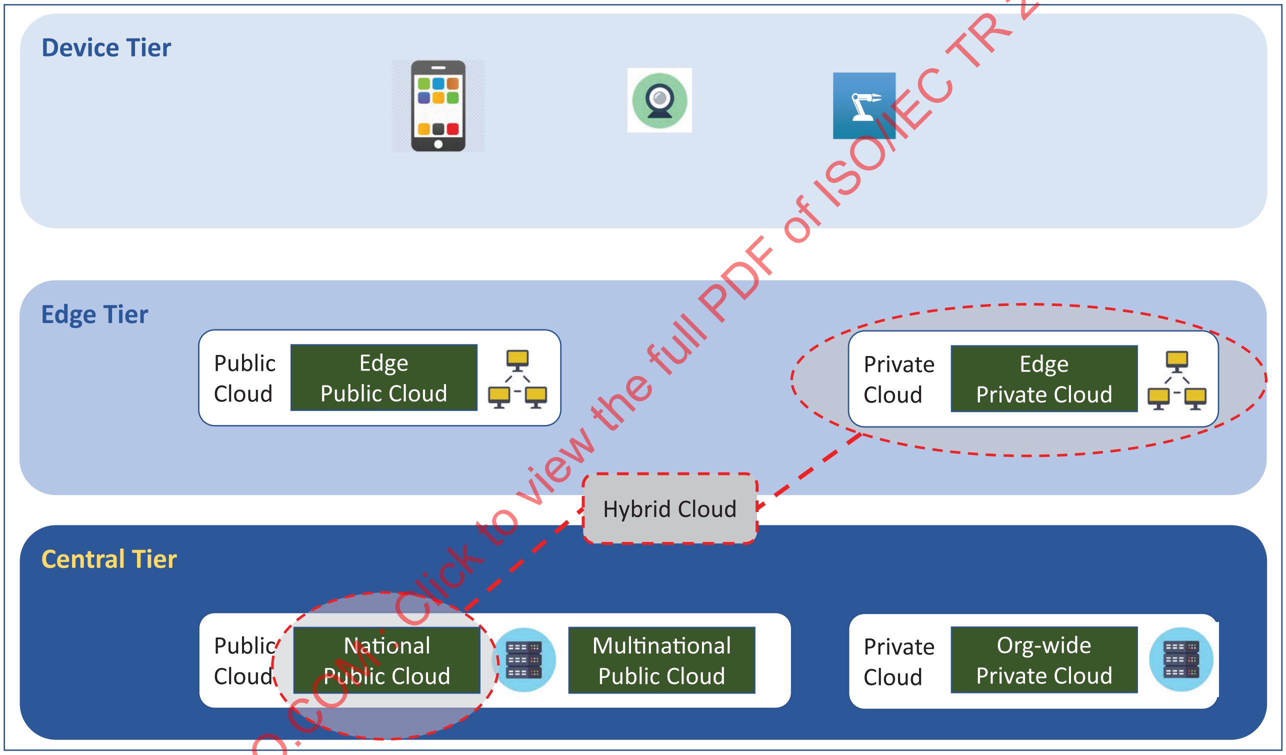

Figure 2 — Relationship of edge computing to cloud computing

The central tier can be implemented using public cloud services, and these can be implemented using either a multinational public cloud (i.e. multiple data centres in a number of jurisdictions) or a national public cloud (for example, where there are restrictions on where the data can be stored and processed). The central tier can also be implemented using an enterprise-wide private cloud.

The edge tier can be implemented using an on-premises private cloud, physically located to suit the needs of the edge system. However, it is the case that some public clouds make available cloud services on a more physically localised basis, potentially suitable for edge computing scenarios. Compute capabilities in cellular telephone towers are an example. Such distributed public cloud offerings could be used for the edge tier. As an example in this latter case, the device tier nodes could be directly connected using mobile phone networks to nodes running in cellular telephone towers.

As an example, to reduce latency and achieve timing goals, it could be necessary to place control processing on edge tier nodes, where those nodes are physically close to the device tier such as a co-located on-premises private cloud. However, for processing that involves analysis of large volumes of data that arise from many different sources, it is likely that centralized storage is the best approach, allied to the substantial processing power available for analytics software in the central tier implemented using public cloud services. For any given system, it is likely that both types of processing are required and need to be combined effectively.

So, for example, direct control of a manufacturing production line is likely to be placed in the edge tier – particularly where real time responses are required to the arrival of data and events. However, the overall goals of the production line such as the product mix to produce, are much more likely to be decided by applications and services in the central tier, which are analysing a lot of external data, combined with information about business goals. Such goals are then passed down from the central tier to the edge tier for implementation.

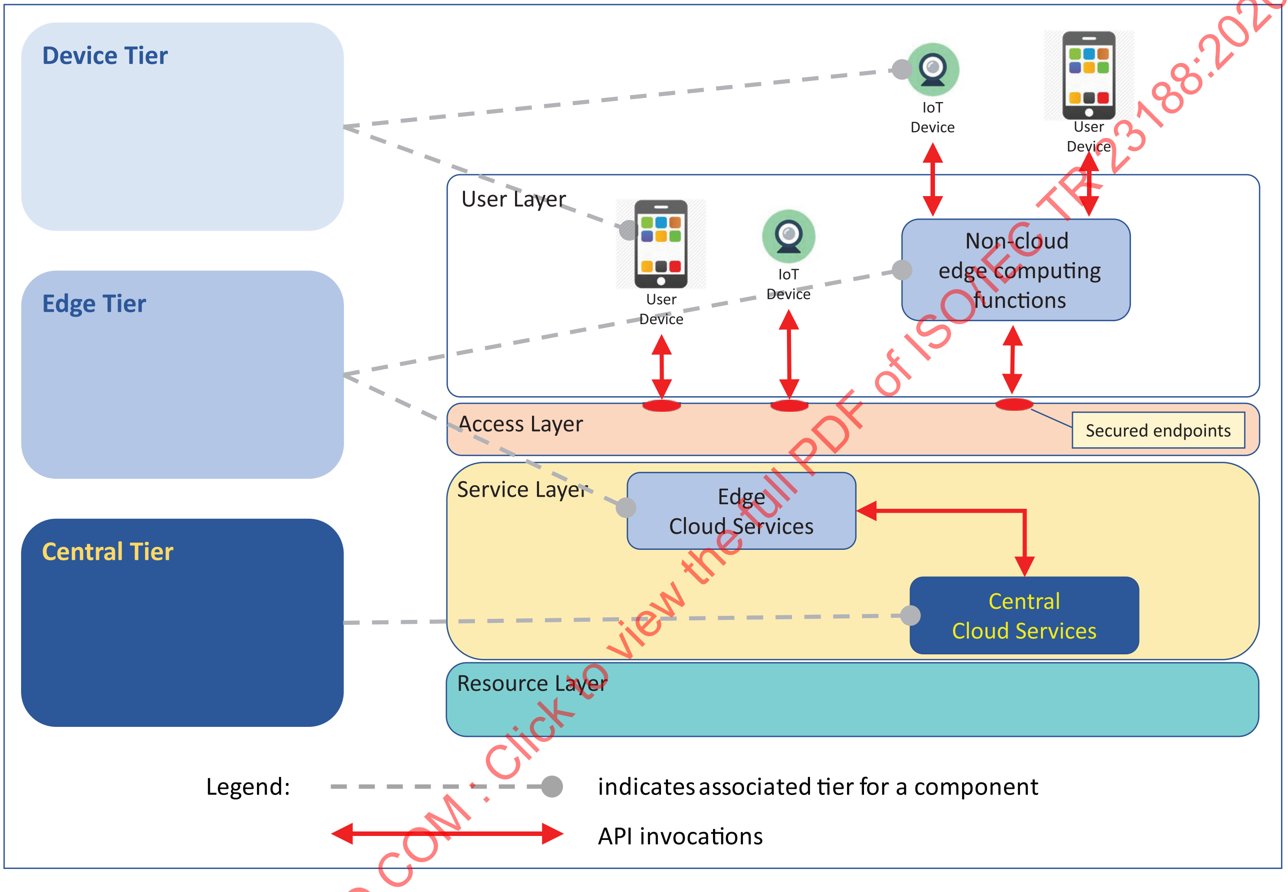

Figure 3 — Relationship of edge computing tiers to cloud computing

Figure 3 aims to show the relationship of edge computing to cloud computing as described in ISO/IEC 17789[5]. It is necessary to appreciate that the edge computing tier model, shown on the left is largely a physical model primarily concerned with placement of nodes. The ISO/IEC 17789[5] functional view shown on the right is a logical model primarily concerned with component relationships. In this view, the devices – and so the whole of the device tier – belong to the user layer of cloud computing, or, if the devices do not use cloud services themselves, they are outside the cloud computing model entirely as shown on the top right. Thus Figure 3 is making the assumption that cloud computing is not being implemented on the devices themselves.

The edge tier is represented by sets of edge computing functions, which represent edge capabilities implemented as software services, and the edge computing functions can be implemented as non-cloud edge computing functions, in which case they logically sit in the cloud computing user layer. Alternatively, the edge computing functions can be implemented as cloud services, in which case they logically sit in the cloud computing service layer. Typically gateway nodes in the edge tier would be implemented as non-cloud edge computing functions. Control nodes in the edge tier could be implemented as either non-cloud edge computing functions or as edge cloud services. Meanwhile, the central tier is usually based on a series of cloud services, typically implemented in a centralized location such as a public cloud computing data centre. Each of the services is used by clients (e.g. user devices or IoT devices) via API invocations to the service interface, whether the services are implemented by cloud services or not.

This indicates that many edge computing systems are effectively implemented by CSCs rather than CSPs – the elements in the user layer are implemented and under the control of the CSC – not the CSP. An exception to this can occur in the case where the cloud computing system is a private cloud deployment which is owned and operated by the same organization that owns and operates the device and edge tiers.

It is important to note that the multi-layer functions of cloud computing (not shown in Figure 3) do span across all the edge computing tiers, including those in the user layer of cloud computing. This implies that capabilities such as security, privacy and management have to span across all the tiers and need to be implemented in a coherent way – otherwise problems are likely to occur. As an important element of security, the endpoints for cloud services present in the access layer are secured to protect the underlying cloud services from attack, as indicated in Figure 3.

It is also important to differentiate cloud computing software technologies from cloud computing itself. For example, the fact that a software component or an entire application can be packaged in one or more container(s) and be deployed and run on one or more computers does not mean that cloud computing has been realized there. To qualify as cloud computing, there are key characteristics such as scalability and elasticity that need to be met (see ISO/IEC 17788[5] and ISO/IEC 22123-1 for cloud computing key characteristics).

However, there are a range of common technologies associated with cloud computing that are often used in edge computing, such as virtualization, VMs, containers and serverless computing (as described in Clause 8 and in ISO/IEC TS 23167).

Some CSPs offer cloud services that have capabilities that are designed to support edge computing. Many of these cloud services are particularly designed to support IoT systems and include capabilities such as:

— device registration;

— device management and operation;

— device communication (e.g. support of lightweight protocols);

— device storage.

5.5 The relationship of edge computing to IoT

Edge computing is a fundamental technology for the implementation of IoT systems. IoT systems are based on sensor and actuator IoT devices that exist in the device tier described in 5.3. Although some IoT systems do not include any cloud computing or edge computing elements, many IoT systems do include an edge tier, containing IoT gateways and control nodes. It is also very common for IoT systems to make substantial use of a central tier, often containing analytic services and management and control systems, commonly implemented using cloud computing.

These different possibilities can be represented as shown in Figure 4:

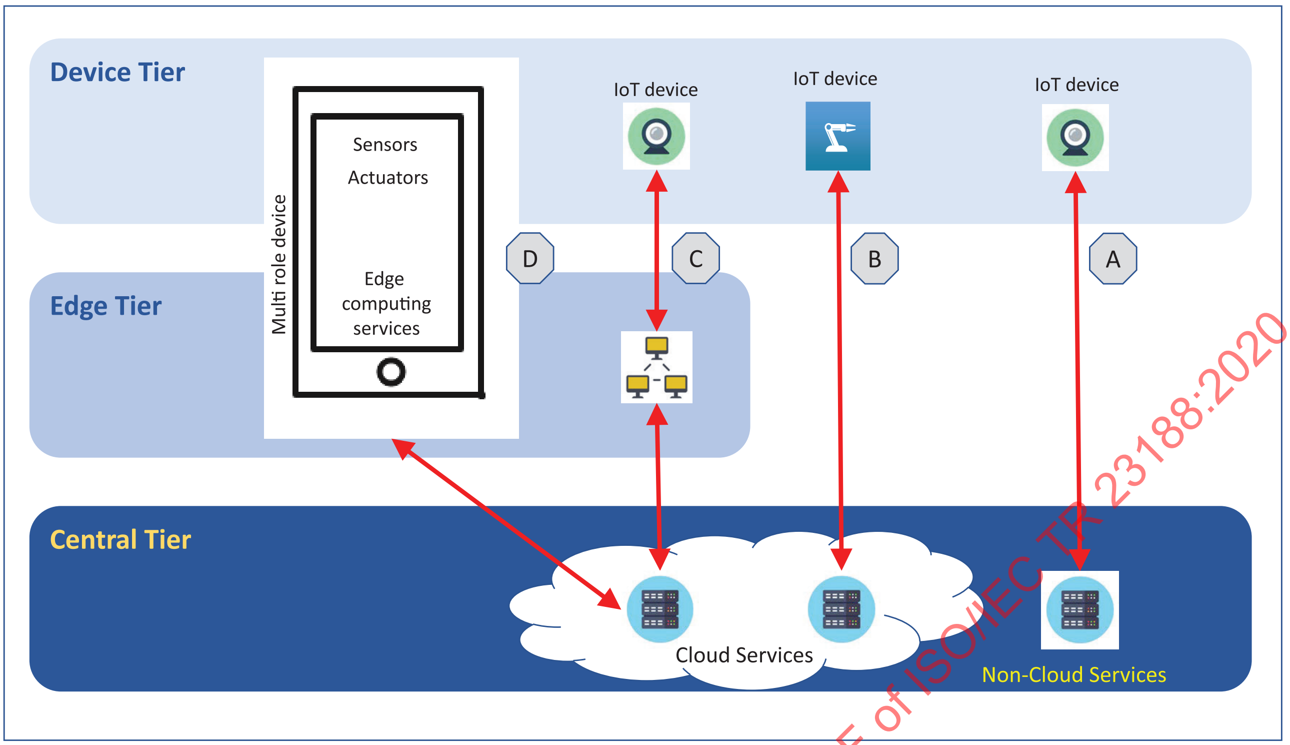

Figure 4 — Relationship of edge computing to IoT

The IoT, edge computing, and cloud computing concepts are independent. While often deployed together, the presence of any one does not imply the presence of all.

The four cases detailed in Figure 4 are as follows:

a) Non-cloud IoT. Some IoT systems do not employ cloud computing or edge computing; IoT devices can be connected to non-cloud central services without any use of edge computing, as shown by “A” in Figure 4. These are out of scope for this document.

b) Cloud IoT. Some IoT devices are directly connected to a central cloud service without any use of edge computing, as shown by “B” in Figure 4. These are also out of scope for this document

c) Edge IoT. Some IoT devices are connected to edge computing services (that might be implemented using cloud services, as edge cloud services) that are themselves connected to cloud services, as shown by “C” in Figure 4.

d) Multirole edge computing. Some devices are multirole devices and can function as both IoT devices and as an edge computing node simultaneously, for example a smartphone as a user interface device with IoT sensors running different apps for different purposes, as shown by “D” in Figure 4. These multiple functions within the device might or might not be aware of each other.

The model of the edge computing tiers presented in 5.4 above therefore describes a general case, with no implication that any of the tiers is always present or required. Each system includes whichever tiers are appropriate to the functions being deployed.

6 Networking and edge computing

6.1 General

Networking is a key element of edge computing. Edge computing involves multiple nodes, which are networked machines, with activities coordinated between the nodes by means of digital messages passed between them over the networks that connect them.

It is typical for edge computing to involve a set of different networks, where the network used to connect a particular set of nodes reflects the nature of the nodes themselves and also the location and environment in which they exist.

The ISO/IEC 30141[2] communications view describes 4 types of network, which are broadly applicable to edge computing:

a) proximity network;

b) access network;

c) services network;

d) user network.

6.1.1 Proximity networks

Proximity networks exist at the edge. Their main task is to connect the nodes in the device tier – the sensors, actuators and user interface devices. Proximity networks connect the device tier nodes to each other, but more commonly connect those devices to nodes in the edge tier, such as gateways and control nodes.

It is commonly the case that proximity networks are local and limited in range – they could also have limited bandwidth. This is driven by the nature of the nodes at the edge – they are commonly low-power devices (in some cases "no-power" devices) that cannot support the technology required for wide-area communications, or they are in locations that make it difficult or impossible to provide wide-area communications.

Proximity networks often use specialized protocols, can involve intermittent communications and might not support the Internet Protocol (IP). Proximity networks often involve very simple, local and low-power hardware. There are different kinds of proximity network in existence. A small set of example proximity networks include IPv6 over Low Power Wireless Personal Area Networks (6LoWPAN), ZigBee, Wireless HART, and Narrow Band IoT. It is common that proximity networks are wireless in nature, to avoid the need for cabling to connect to each node, which could be impractical or too expensive.

6.1.2 Access networks

Access networks connect nodes within the edge tier to each other and to nodes in the central tier. Access networks are typically wide area networks and they also typically support IP. A range of technologies can be employed for access networks, including wired connections (LAN, Broadband, Fiber) and wireless connections (Wireless LANs, Mobile (cellular), Low power wide area networks, Satellite links).

6.1.3 Services networks

Services networks typically connect applications and services running in the central tier. These networks are typically high bandwidth low-latency networks running IP. Services networks typically exist within a cloud computing data centre, although CSPs can support such connections between two or more physically separated data centres.

6.1.4 User networks

User networks connect end-user nodes to the applications and services running in one or more of the tiers of the system. Such end-user nodes can offer human interfaces, or they can represent automated systems with no human interface. User networks are commonly based on the public internet and use IP – and they can use any of the technologies commonly used to carry internet traffic, including both wired and wireless systems. It is increasingly the case that the end-user nodes are either smartphones or devices which use smartphone technologies, with increasing use of wireless network technologies including Wireless LANs and Mobile (cellular).

6.2 Virtual networks

From the point of view of software in edge computing, it is increasingly the case that virtual networks are more significant than the physical networks which connect nodes. Virtual networks, including Software Defined Networks (SDNs) and Network Functional Virtualization (NFV), are a natural partner to both virtualized compute (VMs, containers) and also to virtualized storage. Virtual networks use the capabilities of the underlying networking hardware, combined with various software elements to provide networking capabilities that are tailored to the specific needs of the distributed applications running in an edge computing system.

Virtualized compute environments, such as VMs and containers, involve tight control and virtualization of network resources – including both the endpoints exposed by the virtualized environments and also the target network endpoints used by the software running within the environments. The capability of running multiple VMs on one system, or multiple containers on one operating system, clearly requires mapping of each of the network endpoints exposed by the software to actual endpoints in the containing system, simply to enable the sharing of the system without resulting in clashes. The deployment of both VMs and containers requires configuration to deal with these issues.

It is also commonly the case that both VMs and containers are used in distributed environments. Multiple instances of the same software can run on different systems and those systems can run in different physical locations. Applications are also commonly divided into multiple independent components (services, microservices), and these components can each run in separate locations. Ideally, the actual locations of the components are hidden from the software – application components need to communicate with each other seamlessly wherever they are running. In addition, it is best for security reasons if the application components can only communicate with each other – external communication is usually carefully controlled through specifically designated external endpoints.

The ideal approach to networking for these software architectures is that networking is effectively defined at the application level – a virtual network that is used only by the application components and which transparently spans all the locations in which application components are running. The implication is that each application has an associated virtual network – and that, as with virtual compute environments like containers, each virtual network is isolated from other networks. More detail on virtual networks and their relationship to virtual compute environments and to virtualized storage is provided in ISO/IEC TS 23167.

There is a security aspect to virtual networking, important in supporting distributed applications. With non-virtualized networking, endpoints are exposed on each system where a software component is run – and each endpoint must be fully secured since in principle each endpoint is publicly accessible. With virtual networks, it is possible to define a network only visible and usable by the components of a specific application, wherever those components are running. Other software, not granted access to the virtual network, cannot use that network and cannot get access to any of its endpoints. Put another way, the virtual network itself takes on the burden of securing the endpoints within the network and only endpoints deliberately exposed outside the virtual network need to have additional security applied. Equally, software components connected to the virtual network can be (by design) limited to seeing and using only those services attached to the same virtual network.

A variety of technologies are available to implement virtual networks, some designed for specific types of environment. These include:

— VxLAN: an overlay network technology, specified in IETF RFC 7348[9];

— Kubernetes networking;

— Container networking systems, such as Calico[11] and Weave Net[12].

7 Hardware considerations for edge computing

7.1 General

The hardware used for nodes in edge computing can vary significantly. The nature of the nodes in edge computing has a major impact on system organization, since the placement of capabilities in the system is often dictated by the hardware capabilities of the nodes.

7.2 Hardware capabilities

In terms of the capabilities of nodes in edge computing the key factors are:

— the amount of compute power;

— the nature of the hardware processors involved;

— the amount of runtime memory;

— the amount and type of storage;

— the network bandwidth and latency;

— the network type and its ability to integrate into a larger system;

— security capabilities, e.g. chipset with TPM; software update supported.

For nodes in the device tier, cost is often a key factor since there can be many nodes and keeping their cost down can be a significant factor in the overall cost of the system. Cost can include not only the hardware of the device, but also the impact of providing sufficient electrical power to run the device. Cost and the physical environment of the devices can also influence the network technology that is used. Another factor is the requirement to maintain devices that can be placed into locations that are difficult and/or expensive for maintenance personnel to reach. This means that physical maintenance needs to be kept to a minimum.

It is also the case that the cost of edge nodes of a given capability continues to decrease over time while the capabilities of nodes of a given cost continue to increase.

An example is the evolution of the Raspberry Pi system, a low cost system often used in edge computing. The original Pi 1 had a single core 700 Mhz ARM11 processor with 128 Mb RAM while the Pi 3 has a quad core 1,2 Ghz ARM Cortex-A53 with 1Gb RAM which can have up to 10 times the performance of the Pi 1. The expansion of RAM size is probably almost as important as the higher performance, in that it enables more sophisticated software to run on the device. This evolution occurred between February 2012 and February 2016 when the advanced capabilities of the Pi 3 became available at the same cost as the original Pi 1.

With the increasing need to implement sophisticated applications (e.g. surveillance, access control, anomaly detection) on the edge, nodes in the edge tier become increasingly powerful to offer intelligence. Data transformation, advanced analytics and local decision-making on edge nodes has become more common. Traditional CPUs or microcontrollers are complemented by heterogeneous accelerators, such as GPUs, FPGAs or ASICs of various types, to enhance the performance and efficiency of intensive and complex processing (e.g. inference). Moreover, the hardware architecture is tailored to specific application needs and the constraints of the environment. For instance, the edge nodes for surveillance can have an architecture optimized for image or video processing, with a form factor suitable for deployment on the road, at the gate or in the building.

Another important set of hardware capabilities for an edge computing node are sensor and actuator capabilities.

A sensor is an IoT device that measures some property of a physical entity and outputs digital data representing the measurement that can be transmitted over a network (ISO/IEC 20924[8]). A simple example is a thermometer which measures the temperature of the air in a room where the output might be a message containing the temperature represented as a number in degrees Celsius. In other cases, the output digital data can be distantly related to the input from the physical environment and can be output with a significant time delay, depending on processing of the data within the node. For example, the recognition of the identity of a vehicle number plate from a surveillance camera device can be based on the input video of a scene where the output is a message containing a string representing the recognised number plate. In some cases, the physical entity that is measured is the node itself. For example, a thermometer in a node can measure the node's temperature, or an accelerometer in a node can measure the movement of the node.

An actuator is a device that accepts digital inputs and which acts on (i.e. changes) one or more properties of a physical entity on the basis of those inputs (ISO/IEC 30141[2]). This can be as simple as switching on or off an indicator light or as complex as instructing a manufacturing robot capable of manipulating physical objects.

Actuators can require control software loops, where inputs from one or more sensors are processed by control software which then issues commands to the actuators. Control software loops are often intimately associated with the related sensors and actuators and operate with real time deadlines (see Clause 14) to ensure that the actuators perform appropriate actions on physical entities.

It is also notable that there are classes of complex devices that combine sets of sensors, actuators with substantial processing and communications capabilities and also with human user interfaces – these types of complex devices are considered further in Clause 15.

8 Software technologies for edge computing

8.1 General

Software is increasingly a key element in edge computing when the capabilities of a node in an edge computing system are defined and implemented by the software that exists and runs on that node.

However, the nature, organization and capabilities of software vary substantially from node to node. Partly, this is influenced by the capabilities of the nodes themselves. Partly it is influenced by the choice of particular software technologies. Both the capabilities of the nodes themselves and the available software technologies are advancing over time.

There are various classifications of software that are involved in edge computing, described in 8.2. Some of the significant software technologies used in edge computing are described in 8.3.

8.2 Software classifications

8.2.1 Firmware

Firmware is a class of software that is integrated with the hardware of a device and which provides for low-level control of that device's hardware. Firmware is typically held in non-volatile memory within the device such as ROM, EPROM or flash memory. Firmware typically has limited capabilities which are specific to the hardware of the device.

Simple devices which have limited, fixed functionality could contain and run only firmware. In more advanced devices, the firmware could form the foundation for more complex software to run on the device, for example, supporting platform software.

For some types of device, it is not possible to change the firmware of the device after manufacture. For other types of device, it is possible to update the firmware, for example to fix bugs or to add capabilities to the device. Where supported, the method of updating the firmware can vary from physical replacement of the memory storing the firmware, to the update of flash memory contents via a special procedure.

In general, it is desirable that firmware can be updated, especially to address any security vulnerabilities that are discovered in the firmware. In an edge computing system, in addition to manual local updates, it is also desirable that the firmware update can be performed remotely over the network, with appropriate security controls to prevent malicious updates to the software.

8.2.2 Platform software

Platform software is software that provides an environment in which other (higher level) software is executed.

The most typical form of platform software is the operating system (OS) such as Linux, Microsoft Windows and Azure Sphere, Android, RTLinux (and a large set of similar real time operating systems).

Platforms can also include virtualization software such as virtual machines (VMs) and containers.

In other cases, platform software includes a runtime environment which is necessary to execute software components designed for that runtime environment. Examples of general purpose runtime environments include Java VM, node.JS and Microsoft .NET Common Language Runtime, while an example of a more specialised runtime environment is NodeRED[6]. Some of these runtime environments include a substantial stack of software offering a wide variety of capabilities.

In general, one of the aims of platform software is to provide a set of capabilities which can be used by the higher level software that runs on the platform. This permits the simplification and reduction of the amount of code in the higher level software.

Note that platform software can provide one or more services (as described in 8.2.3) as a means of providing capabilities to higher level software. Some of these services can relate to the use of the platform itself, such as lifecycle services.

Note that the term "platform" is subject to over-use and can be misapplied. A useful discussion of the various types of software platforms can be found in the article "The 9 types of Software Platforms"[7].

8.2.3 Services

Services are software components that offer specific capabilities to other client software via one or more defined interfaces. Services operate independently of the client software that uses them, for example, operating in their own OS process.

The capabilities offered by software services can be of almost any type. Examples include encryption/decryption, video processing, artificial intelligence, analytics.

8.2.4 Applications

Application software is a software component that is designed to perform a coordinated set of tasks for some particular purpose or goal. Application software can directly serve end-users and in such a case, the application is likely to offer some form of user interface both to display information to the end-user and also to receive input or commands from the end-user. In other cases, application software can operate autonomously without a direct end-user interface.

Application software can make use of software services and also make use of capabilities provided by platform software.

An example of application software within edge computing is the applications used to control a smart building that ensure that the facilities within the building operate to serve the users of the building, operate efficiently and alerting maintenance staff where problems occur.

8.3 Significant software technologies

8.3.1 General

Increasingly, edge computing is taking advantage of modern software technologies to deal with the development, distribution and execution of software across all the different tiers and nodes.

Many of these technologies had their origins in cloud computing, but they are now being applied to some edge nodes, where the capabilities of the edge nodes are sufficiently high. The thinking behind the use of these technologies is "right placement" of the software for the task in hand, taking into account the processing and storage capabilities of the nodes and the bandwidth and latencies associated with the networks which connect them.

The technologies described in this clause are described at more length and in more detail in ISO/IEC TS 23167. This clause concentrates on the application of the technologies to edge computing.

8.3.2 Virtual machines

A virtual machine (VM) is an isolated execution environment for running software that uses virtualized physical resources. In other words, this involves the virtualization of the system where the software within each VM is given carefully controlled access to the physical resources to enable sharing of those resources without interference (from ISO/IEC 22123-1:—, 5.5.1). Multiple VMs can run at the same time on one hardware system. The purpose of VMs is to enable multiple applications to run at the same time on one hardware system, while those applications remain isolated from each other. The software running within each VM appears to have its own system hardware, such as processor, runtime memory, storage device(s) and networking hardware.

Each VM contains a complete stack of software, starting with the operating system and continuing with whatever other software is required to run the application(s) that execute within the VM.

The software that runs in a VM is stored and distributed in the form of a VM image. To run the software, the target node runs a hypervisor and any associated control and management software.

Since a VM contains a complete software stack, this means that the VM can run almost anywhere. However, the completeness of the software stack often means that the VM image is of substantial size, both as an image and also at runtime. This can make distributing the VM images slower and more bandwidth intensive. It also means that (relatively) few VMs can run on a particular system due to the resources they consume (especially runtime memory), which is often a serious consideration for edge nodes since they are typically relatively small systems.

For a complete and detailed description of VMs and associated software, see ISO/IEC TS 23167:2020, Clause 6.

8.3.3 Containers

A container is an isolated execution environment for running software that uses a virtualized operating system kernel (termed the host operating system). Virtualized operating system kernel means access to the resources of the operating system kernel is mediated and the software within the container only gets to see and use a carefully controlled and limited version of the host OS resources (from ISO/IEC 22123-1:—, 5.5.2). Multiple containers can run at the same time on one hardware system. Containers enable multiple applications to run at the same time while those applications remain isolated from each other.

Each container typically contains application software and the minimum stack of software on which it depends. This stack does not include the operating system kernel, which is provided by the host OS.

The software that runs in a container is stored and distributed in the form of a container image. To run the software, the target node runs a container runtime environment and any associated control and management software. Container images can be relatively small, depending on the size of the application code itself and on the number and size of its dependencies. Container images can also be layered, so that common pieces of software such as runtime environments (e.g. node.js, Java) can be shared by multiple containers on a particular system.

By comparison with running applications in VMs, containers use less resources (less memory in particular) and can be started and stopped relatively quickly as needed.

For a complete and detailed description of containers and associated software, see ISO/IEC TS 23167:2020, Clause 7.

8.3.4 Serverless computing

Serverless computing is a cloud service category in which the CSC can use different cloud capabilities types without the CSC having to provision, deploy and manage either hardware or software resources, other than providing CSC application code or providing CSC data.

One common form of serverless computing is Functions as a Service (FaaS). FaaS is a form of serverless computing in which the capability used by the CSC is the execution of CSC application code, in the form of one or more functions that are each triggered by a CSC specified event.

Serverless database is another form of serverless computing, in which the capability used by the CSC is a database, where the database is provisioned, managed and operated by the CSP and its functions are made available via an API. The allocation of storage resources is managed by the CSP.

For a complete and detailed description of serverless computing, see ISO/IEC TS 23167:2020, Clause 8.

8.3.5 Microservices

Microservices architecture is a design approach for building cloud native applications, which is also useful for building applications suited to edge computing.

For an application built using microservices architecture, the application is divided into a series of separate processes termed microservices, which are deployed independently and which are connected by service interfaces. Each microservice is designed to implement some specific area of function, say a particular technical capability of a particular business process. Each microservice can be developed, tested, operated and updated independently.

A full description of microservices is given in ISO/IEC TS 23167:2020, Clause 9.

9 Deployment models and service capabilities types and service categories for edge computing

Cloud computing has a variety of deployment models, cloud capabilities types and cloud service categories, as described in ISO/IEC 17789[4] and ISO/IEC 17788[5]. Similar models, types and categories apply to edge computing.

9.1 Deployment models

The nodes in the device tier, the edge tier and the central tier can be owned and managed by a single organization. This is equivalent to a private deployment model across all the tiers:

— private device deployment;

— private edge deployment;

— private central deployment.

However, other deployment models are possible for each tier and the deployment model for one tier does not have to be the same as that of other tiers. In each tier, the following deployment models can exist:

— private deployment

  where the deployment is either owned and managed by a single organization, or where a provider makes the resources available exclusively to a single organization (i.e. no sharing of resources);

— public deployment

where the deployment is owned and managed by a provider who makes the resources available to a wide range of customers (i.e. resources are shared);

— community deployment

where the deployment is owned and managed on behalf of a group of organizations who have a relationship with one another and a shared set of requirements (i.e. the resources are shared amongst the community);

— hybrid deployment

where a given tier has a combination of the other deployment models.

As an example, telecommunication organizations are making available edge nodes as a service where these nodes are based around the widely distributed network hardware operated by these organizations. For example, nodes with compute and storage capabilities can be available in mobile phone network base stations.

9.2 Service model capabilities types

As for cloud computing, the capabilities provided by any node in any of the tiers can be one of the three:

— infrastructure

where the user of the node can provision and use processing, storage or networking resources;

— platform

where the user can deploy, manage and run user-created or user-acquired applications using one or more execution environments supported by the node;

— application

where the user can make use of one or more applications provided by the node.

9.3 Service categories

In cloud computing, a service category is defined as "a group of cloud services that possess some common set of qualities".

As for cloud computing, edge computing services can be offered in any of the tiers and can be any one of many different service categories, of which examples include:

— analytics;

— image processing;

— entity tracking;

— control processing;

— content distribution;

— IPTV;

— platform services such as running containers, object stores, databases.

10 Data in edge computing

10.1 General

Data is a key element of any form of computing. Since edge computing is a form of distributed computing, three aspects inevitably apply to data in edge computing:

— the flow of data between nodes in the system;

— the storage of data on nodes in the system;

— the processing of data by nodes in the system.

These three aspects of data in edge computing are considered in this clause.

10.2 Data flow

In edge computing systems, data is typically generated by sensor devices and also through input to user interface devices. Other data can flow into the system from outside sources and can be held on nodes in any tier of the system. Data can also be produced within the system as the result of processing activities such as analytics. This can include derived data resulting from the use of the system as described in ISO/IEC 19944[22].

Data flow can take place between two or more nodes within one edge computing tier, or it can take place between nodes that exist in different edge computing tiers. The essence of the data flow is that the data is made available to storage and processing resources on the receiving node(s). It is useful to place the data flow in the context of the model of edge computing described in Clause 5 – as shown in Figure 5. It is notable that data flows can be both between nodes in one tier (e.g. between different devices in the device tier) and also between nodes in different tiers (e.g. between a device in the device tier and an edge tier node).

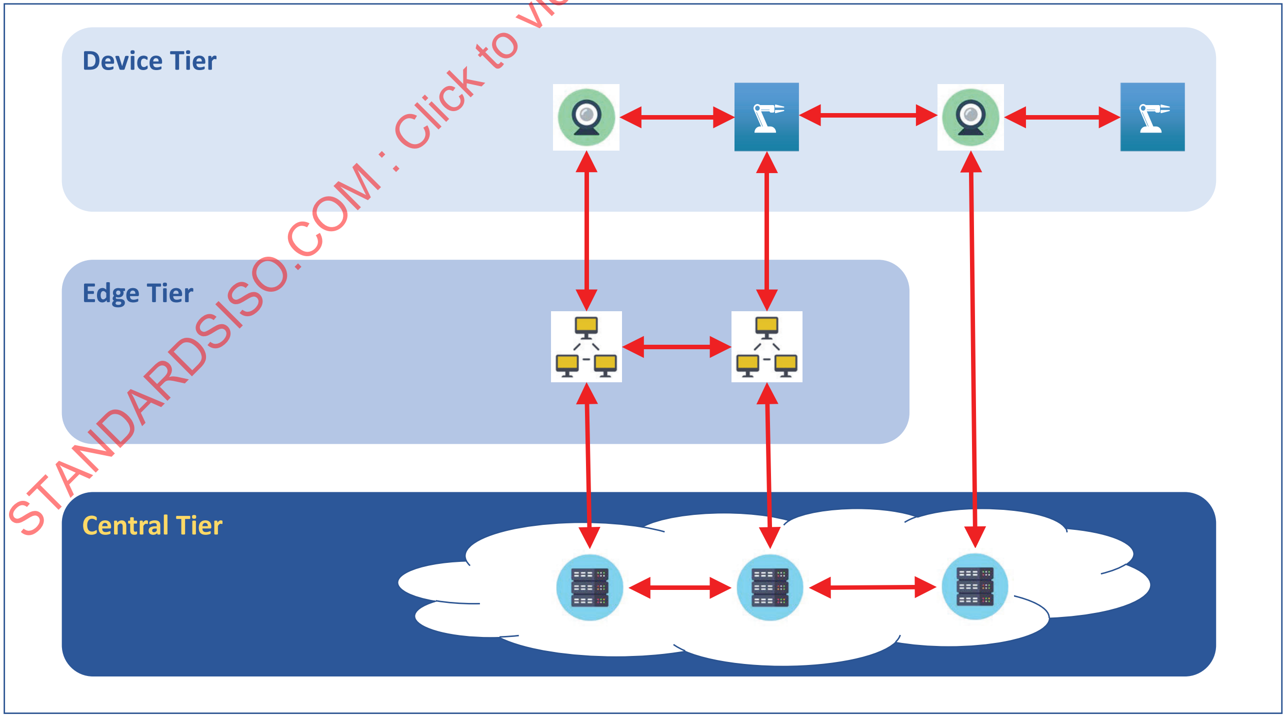

Figure 5 — Data flows in edge computing systems

One purpose for data flow is that there can be insufficient storage capacity available on the source node. For example, it is often the case that nodes in the device tier have limited storage capacity. In such cases, the data must flow to other nodes for long term storage, typically in another tier. Similarly, the data could require software analysis that takes more processing power than is available on the source node. In this case it is necessary that the data flows to node(s) where sufficient processing power is available. Data flow is also necessary to support resilience and availability objectives when data is stored in multiple redundant locations or datasets are replicated on multiple nodes in the system.

Another important purpose for data flow is to aggregate data coming from multiple sources. Some types of data analysis are only possible when combining data from multiple sources, such as multiple sensors, or where current data is compared with historical data. Data aggregation is a major consideration in the placement of data. It is often the case that data flow from the device tier to one of the other tiers is necessary since that other tier is capable of receiving data from a wider span. The edge tier might aggregate data from a set of devices that are physically close to the edge tier nodes such as from within a single building while central tier nodes can aggregate data from a set of edge tier nodes that are physically distant from one another, such as multiple buildings across a city or buildings in different cities.

Data flow inevitably leads to the consideration of networks, since it is over the networks that the data flows. As described in Clause 6, there are various different kinds of networks associated with edge computing systems. The networks can vary in their bandwidths and also in the latency associated with moving data over those networks. The bandwidth and latency can have considerable impacts on system design, as described in Clause 14 and can influence the placement of both storage and of software components.

Network bandwidth limitations can be addressed by means of techniques that limit the amount of data that flows. One possibility is to flow data only when a significant change takes place. Compression of the data is another technique that can reduce the volume of the data to flow. Another possibility is to flow data in digest form or where pre-processing has taken place to reduce the volume of data. An example of pre-processing is dealing with facial recognition from a video stream, where the video stream is processed locally to extract the essential characteristics of a face, which involve much less data and which can then flow across the network to node(s) where the characteristics can be compared against a potentially large database of known faces.

An additional consideration applies where data is used at or near the edge, but where that data has a source that is distant from the edge. The case of network caching of data (especially audio and video data) ready to use at edge nodes (often user interface devices) has been common in association with web serving for a long time. The concept of caching can apply to any centrally held data that needs to be used at or near the edge for other use cases such as augmented reality and virtual reality. Data caching can improve delivery speeds and reduce network traffic.

A specific example of the use of Edge functions to mitigate network bandwidth usage is in video streaming services such as non-Internet IPTV and internet-based video streaming services. IPTV systems and internet Content Distribution Networks (CDNs) commonly use Edge-based caches of popular video content placed as close as possible to the consumer (e.g. in the local telephone exchange or cable TV head-end). This approach reduces bandwidth usage in the core network, and can also enable advanced video features such as fast channel change between live IPTV channels (e.g. ETSI TS 102 034 V2.1.1 (2016-04) – Annex I[24]).

However, there are also good reasons why data flow can be undesirable. One case is where the data involved has privacy and/or confidentiality concerns. Flowing such data can expose the data to the risk of falling into the hands of unauthorized people. Another case is where the data concerned is subject to laws and regulations that restrict either the movement of the data or the purposes for which it can be processed. In other cases, data flow can be wasteful of limited network bandwidth.

Describing the flow of data across the 3 tiers of edge computing is especially useful when there is a need for transparency about where data is generated, where it is processed and where and how it is used. This may be necessary to address concerns about security, privacy and rights and responsibilities involving data and its value. For more information about data flows, data categories and data use statements see ISO/IEC 19944[22].

10.3 Data storage

Data can be stored on any node in the edge computing system. The amount of storage available varies from one node to another. Generally, there is less storage volume available on device tier nodes, due to size, cost and power consumption reasons. Edge tier nodes often have more storage space available, while central tier nodes typically have very large amounts of storage capacity ("unlimited capacity" is often said of centralized public cloud computing data centres).

The availability of storage capacity can influence system design, with the tendency to store large volumes of data where there is matching capacity, such as in the central tier.

One good reason to store data near the edge is to provide for autonomy. There can be a concern that communication to distant resources, such as those in the central tier, might be subject to interruption. If processing in the edge tier and device tier becomes dependent on these more distant resources, that processing is likely to fail if communication is disrupted. This can have undesirable consequences. An example is the case of an edge computing system supporting a production line in a factory. Failure of the communications to a central tier cloud service can have the effect of stopping production, which is very costly. As a result, such edge tier systems are designed to be autonomous and store the data needed close to the edge to ensure continued operation and use the central tier for capabilities whose failure does not disrupt production.

However, storing data in a single location has the downside of vulnerability to point failures. The failure of a node could involve the loss of all data stored on that node. As a result, replication and backup of data is desirable, at least for any data whose loss could impact the operational effectiveness of the system. Ideally, such replication and backup is built into the design of the system and permits automated recovery after a failure occurs.

While simpler or older systems might use storage devices directly, more sophisticated and advanced systems use one or more storage services to handle the storage of data, as described in ISO/IEC TS 23167:2020, Clause 12. Such storage services can virtualize storage capacity and enable the transparent provision of capabilities such as replication and backup. In addition, storage services can offer enhanced features such as database capabilities, which can better suit the software running on the nodes concerned than raw storage devices. ISO/IEC 17789[4] defines a functional component as a functional building block needed to engage in an activity, backed by an implementation. Data storage is described as resource layer functional components in ISO/IEC 17789[4]. The resource layer functional components include resource abstraction and control and physical resources.

Virtualization of storage permits a flexible, policy-based approach to the locations used to store any particular piece of data. For example, storing frequently used data locally to where it is being used, while lesser used data can be transparently moved to a location where there is greater storage capacity.

10.4 Data processing

Data processing by software is key to any system since it defines what the system does. In an edge computing system, the key questions are what data processing is taking place and where that data processing is taking place.

For cloud computing, data processing is described as services layer functional components in ISO/IEC 17789[4]. The services layer functional components include service capabilities and service orchestration. The service capabilities functional component consists of the necessary software required to implement the service. The service orchestration functional component provides coordination, aggregation and composition of multiple service components in order to deliver the cloud service.

There are two main types of data processing taking place in an edge computing system – analytics and control. Analytics processing is examining data, potentially from multiple sources, to produce results of interest. Control processing is examining data about the real world with the intention of controlling activities in the real world. Analytics and control processing can exist independently, or they could be combined, with the analytics producing results that help the control processing make decisions.

An example of combined analytics and control processing is where monitoring data about some running machinery is analysed in real time, looking for evidence of faults. Where one or more faults are detected, the results are passed to control software that can take action to deal with the fault concerned. For example, the control software can shut down the machinery concerned or make (automated) adjustments to correct for the fault.

Another example involves the use of a mobile device serving the needs of a human user. A smart phone is a typical example of such an edge tier device. To give a good user experience, it is necessary to conduct a significant part of the processing on the mobile device, sending requests to other nodes only where necessary, for example to gain access to up-to-date information from external sources. A smart road map application which can provide real time information on traffic conditions is an example of such a design, where the maps and the control over the view presented to the user are handled locally to provide rapid response to the user, while updates on the traffic conditions flow from a central node that has access to information flowing in from a wide variety of sources.

Where the processing takes place can be a complex question for edge computing systems.

It is often the case that control software is located near the edge either in the device tier or (more commonly) in the edge tier. The location is chosen for reasons of latency and for reasons of autonomy since more distant locations might not be able to react quickly enough to meet the time demands of the system and such locations might be more prone to failures in communications links.

However, edge computing often involves a hierarchy of processing, with software layered through the different tiers, depending on the computing resources available, the volume and location (or origin) of necessary data and the communication links involved.

Layered software involves dividing up the required capabilities into separate software components that operate on different nodes in the system and which cooperate to produce the required results. The different software components can be designed to accommodate the characteristics of the nodes on which they are intended to operate such as dealing with relatively low compute power and limited memory for edge tier devices, while placing software components requiring high compute power and more memory onto central tier systems.

In addition, some processing could require access to multiple different sets of data where those different sets of data have different origins. The origins and volumes of the data involved can affect the placement of the processing. The more different origins and the higher data volumes tends to favour a more centralized location for the processing, since flowing the required data to the (multiple) processing locations in the edge tier can become problematic.

One approach to data processing in edge computing is distributed analytics and distributed queries. In this case, the data in the system is not centralized, but is distributed or "sharded" across many nodes in the system, with different pieces of the overall dataset in different nodes at different locations. Analytic and/or query software components are then also run on these distributed nodes, each node producing a partial result, with the partial results being combined to produce a complete result. While such distributed analytics can be done by software components that are fixed on particular nodes, it is increasingly the case that the software components are designed for a virtual distributed environment, as described in Clause 12. Such software components are then dynamically deployed to the nodes holding the relevant data and executed on demand.

11 Management of edge computing

11.1 Management and orchestration fundamentals

Management and orchestration of edge computing is a significant challenge due to the distributed nature of edge computing systems and the presence of considerable heterogeneity of components across these systems. Orchestration is the automated arrangement, coordination and management of the set of nodes and the related set of software components and data elements that make up the system

Heterogeneity applies to the nodes themselves, especially in the device tier, but also in the edge tier. The node hardware can vary substantially, as also can the software running on the node. Heterogeneity also applies to the networks connecting the nodes. The proximity networks connecting the nodes in the device tier to each other and to nodes in the edge tier can use many different technologies and protocols (see ISO/IEC 30141[2] for further information about proximity network technologies).

One of the major areas of management of edge computing systems is termed operations, which covers collections of capabilities that deal with:

— *Provisioning and Deployment*: Capabilities for lifecycle management for a component including configuration, onboarding, registration, asset tracking, deployment, deactivation and removal.

— *Management*: Capabilities for actively managing components, enabling commands to be issued from a management system that are directed at the various components making up the system.

— *Monitoring and Diagnostics*: Capabilities that enable the continuous gathering of information from the components of the system about their activities and behaviour. The major intent is to ensure that all components are operating correctly and that where deviations from correct behaviour occur, these deviations are identified rapidly, that the cause(s) are identified, alerts generated and where possible, corrective actions taken.

— *Prognostics*: Capabilities using predictive analytics whose main goal is to identify potential issues before they occur and recommend actions to mitigate them.

— *Optimization*: Capabilities that aim to improve reliability and performance of the system, reduce energy consumption, and increase availability as the system is used.

One aspect of many edge computing systems is that such systems often involve large scale where many components are running in many locations. System management for edge computing systems needs to address the scale of the edge system and a key aspect is for a high level of automation of the operations capabilities. This applies to initial installation and deployment, through upgrades and fixes. A further aspect supporting automation is that management processes are distributed and capable of executing autonomously if necessary, to both deal with any outages of communication from the edge tier to central tier and also to provide more timely reaction to events as they occur.

An aspect of dealing with the scale of edge computing systems is that there is a mechanism that keeps track of all the components of the system, both physical components (nodes, networks) and also software and virtual resources. The management capabilities need to know which hardware components are present, how they are connected, what software is in use where and its version level.

The design of an edge computing system aims to address reliability, availability and serviceability ("RAS") as a support for management of the system. Reliability concerns can be addressed for example by using redundant configurations, avoiding single points of failure. Alternatively, components with greater inherent reliability are used. Serviceability covers hardware, software, networking and data components. Hardware components need to be capable of being swapped out and this also implies ease of access to the component. Software components need to be upgradeable whether the software is firmware, platform, service or application. Software needs to be autonomous as far as possible, i.e. upgrades are automated and require minimal intervention. Data needs to be replicated unless the data is truly disposable and its loss would have no impact on the system. Data, including system configuration, needs to be capable of being restored to a system that has been upgraded or swapped.

Infrastructure management is an essential base capability, managing all the nodes in the edge system, including device management as far as possible. It is likely that some of the nodes and devices are remote from the location of the management system and the administrators, so that remote asset management is necessary. It is desirable that nodes support both in-band and out-of-band management capabilities.

In-band capabilities refer to those capabilities that are available when the node is up and running, while out-of-band capabilities are those which are available when the node has been stopped. Out-of-band capabilities might include features such as the ability to stop and restart the node, or to perform a firmware upgrade, which might involve a reboot of the system and the stopping and restarting of the software which normally runs on the node.

11.2 Management plane, control plane and data plane

Given that an edge computing system fundamentally involves distributed computing, with sets of networked nodes communicating with each other to achieve the goals of the system, it is valuable to consider management and control of edge computing systems in terms of capability planes. The concept of capability planes is derived from networking[27][28].

Note that "control" in the sense meant here is not the same as a control system managing the behaviour of actuators in an edge computing system.

It is desirable that all components in the edge computing system provide management and control interfaces. The edge system can be considered as being divided into 3 capability planes, dedicated to separate capabilities:

— *data plane*, which is used for communication between system components for the purposes of the applications and services which embody the business capabilities of the system and through which data is carried in the system.

— *control plane*, which is used for direct control of deployed capabilities where those capabilities relate to the data plane. These capabilities can be automated or can be exercised by one or more of the roles associated with the system, including end-users. It is desirable for control plane capabilities to have some autonomy and continue working even when unable to communicate with other parts of the system.

— *management plane*, which deals with overall management of the system involving capabilities such as the deployment of code and policies and the monitoring of components of all kinds. The management plane can make changes to the control plane. The management plane often involves a centralized set of capabilities for the entire system, dealing with the issues of scale described in 11.1.

Note that these capability planes are not layers and where they are shown on a diagram, there is no significance to their placement in relation to one another.

Security and privacy form cross cutting aspects (as described in ISO/IEC 17789[4] and in Clause 13 of this document) that relate to the security of the systems and the privacy of the data and which apply across the three capability planes and across the tiers of edge computing. Orchestration and harmonization of security and privacy capabilities across the tiers is essential. Different capabilities exist in each of the three planes. So, for example, the data plane can handle encrypted communication of data, while the control plane can handle the encryption keys required to do this, while the management plane can apply the policies as to which communications need to be encrypted and which encryption scheme(s) to use.

The security and privacy capabilities often include activities that depend on functions implemented in the central tier, e.g. registries, directories, certificates, single sign-on, event logging.

A couple of examples from telecom and from IoT can help explain the three plane organization, as described in Table 1.

Table 1 — Examples of data, control and management plane capabilities

Plane	**Example from telecom** **Relating to a telephone line and the service delivered over it.**	**Example from IoT** **Relating to the use of a heating system for a building, with associated heater units and temperature and energy use sensors.**
Data plane	In a telephone system, the sound of the voices in the conversation is carried over the data plane.	Heater units apply energy to heat a building space. Sensors report the energy use and temperature information.

Table 1 *(continued)*

Plane	**Example from telecom** **Relating to a telephone line and the service delivered over it.**	**Example from IoT** **Relating to the use of a heating system for a building, with associated heater units and temperature and energy use sensors.**
Control plane	Making a phone call by dialling a number is an example of the control plane. The routing of an 800 number to the right physical line is an automated control example.	The operator has controls to turn the heating unit on and off, and to adjust the desired temperature. There can be a frost prevention capability that brings the heating units on if the temperature falls below a set temperature. Each unit has an emergency off button and an automatic cut-out to shut it down in the event of a fire or other fault.
Management plane	Having a new telephone line installed, associating a line with a billing account, and allocating a phone number to it are examples of management plane functions. Some telephone sets such as a smartphone require identification, authentication and authorization before telephone or data service is enabled, typically through the use of a physical or virtual SIM card, or implicitly for landline services. Security requires a subscriber's identity to be verified before they can add a new telephone line or make changes to an existing service.	The organisation can employ a centralised heating management system which can monitor all their heating units, push down revised firmware, set limits on when they can be used, how much energy can be used at a given time and how they can be set by the staff, and ensure the units remain secure from hacking. It might also monitor and predict end-of-life and other faults based on individual and aggregated unit data. Security management for the heating system include access controls, identification of components, certificates, detection of breaches, logging of data, control and management plane events.

The three capability planes can extend throughout all three tiers of the edge computing model as shown in Figure 6:

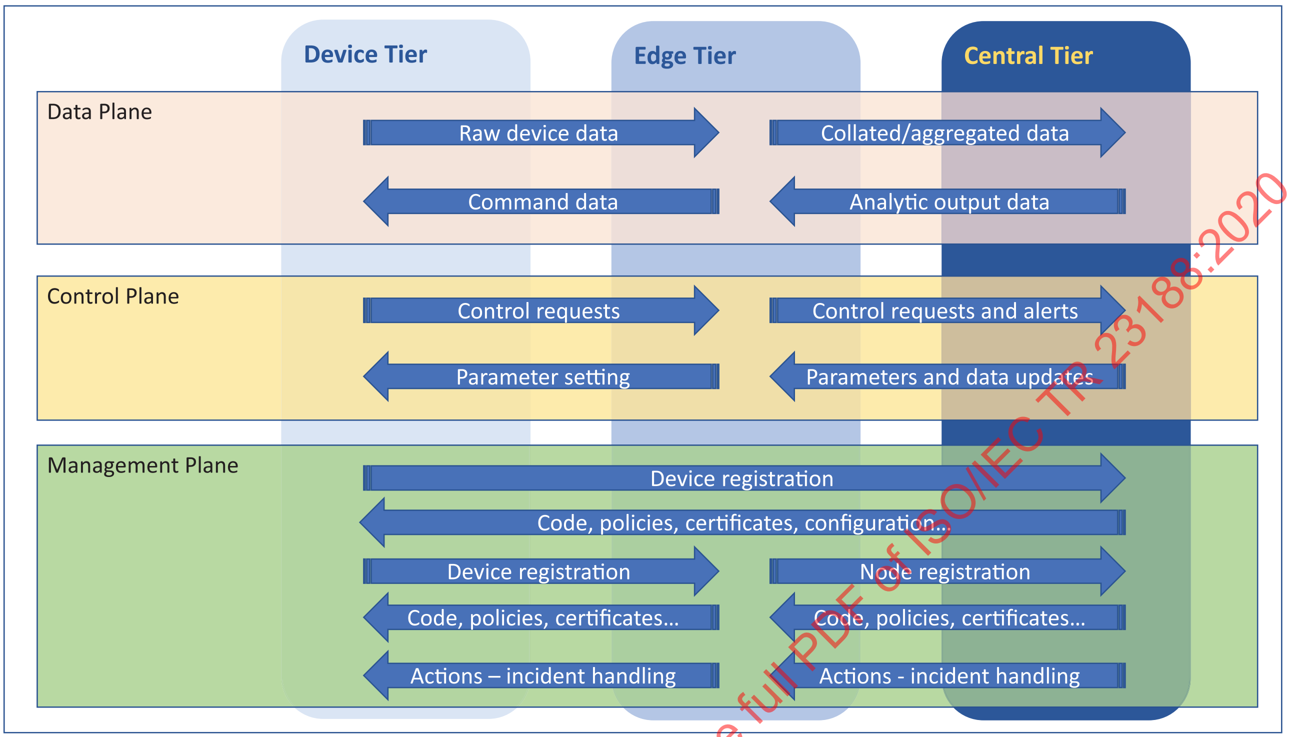

Figure 6 — Relationship of capability planes to edge computing tiers

The arrows within the planes in Figure 6 represent possible abstract information flows within the distributed computing system. The specific information flows shown in Figure 6 are illustrative examples only and are neither prescriptive nor exhaustive of the many possibilities. The direction of the arrows in Figure 6 indicates the general flow of information and does not preclude acknowledgements or other specifics of the protocols used.

The control plane and the management plane need to be isolated from the data plane and from each other, in networking terms. It is desirable that access to each of the control plane and the management plane is strictly controlled and uses additional security mechanisms such as VPN, secure virtual networks and multi-factor authentication.

11.3 Cloud-based management and control of edge tier nodes and device tier devices

11.3.1 General

One area of increasing interest is the use of public or private cloud computing services to provide management and control of device tier devices and edge tier nodes (including but not limited to IoT).

Such cloud-based management neither requires nor precludes the use of cloud computing technologies (such as virtualised software execution environments) in the devices or nodes being managed by the cloud service.

11.3.2 Control of services from a device

In many cases, the device tier node allows a user or application to control some part of the service. The control plane can be used to formulate and make requests of the service (such as making a VoIP call). The cloud service can also use the control plane to deliver alerts and instructions to a device (such as notifying of an incoming call). The data plane can then be used to deliver the service (in this case a VoIP voice connection).

EXAMPLE

A television is connected to a set-top-box which provides access to a cloud-based IPTV service. The viewer is able to request channels, pause the video stream, and control other aspects of the service. The set-top-box connects to an edge tier concentrator node which is physically located in a local telephone exchange. Many commands (such as for popular local channels or stored content) from the IPTV subscriber can be handled directly by the concentrator node. Others (such as requests for more obscure channels) can result in the concentrator node requesting additional service functions from the central tier cloud service.

11.3.3 Management of devices and edge nodes from a cloud service

The management plane can extend beyond central tier computing out into device tier and edge tier, which can be independently owned by the CSC or a third party, and can be located a considerable distance from a centralized cloud computing node.

A CSP can offer management of devices (of whatever kind) from a cloud service. This involves linking the device or edge node and the cloud service in some way, such that the device can receive and will accept management actions directly from the cloud service.

EXAMPLE 1

This approach is already common with mobile phones today, especially in BYOD scenarios where the employer wants to ensure their employees do not put corporate information or systems at risk. A mobile device management (MDM) system works when the employee device is registered with the MDM cloud service, and the control of the device by the end-user (control plane actions) becomes subject to the policies set by the MDM. This can include forcing the installation of phone and app security certificates and patches, virus scanning, delivering and configuring essential corporate apps to the device, enforcing appropriate communications security and, if necessary, removing any corporate information from the device in the event that it is lost or stolen.

EXAMPLE 2

Returning to the IPTV example given in 11.3.2, the management plane is responsible for handling the registration of the set-top-box to the system, verification that the set-top-box is a genuine approved device for the service, and delivery of security certificates that will be needed to access the program stream and to decode encrypted TV channels. The management system instructs the set-top-box on the policies that must be obeyed, such as blocking of private recording of high-quality video outputs. The management system also configures the edge tier concentrator node for matters such as where it should obtain EPG information, security protocols to use, and collects data such as concentrator loading measurements and programme viewing statistics for offline analysis.

While a managed device in the device tier or managed node in the edge tier is certainly managed by the cloud service, it doesn't become part of the managing cloud service, so it remains in the *user layer* of the cloud computing reference architecture. It is also common that such devices or nodes are owned by the service customer or a third party rather than by the CSP.

11.4 Orchestration and maintenance

There is a need for the management system to provide orchestration of the components within the edge computing system. Orchestration primarily involves the coordinated deployment of compute, data and networking components. Where these components are virtualized, as described in ISO/IEC TS 23167, orchestration can involve the use of tools such as container management systems as described in ISO/IEC TS 23167:2020, 7.4.

It is worth noting that edge computing systems, like IT systems more generally, have a need for ongoing development and maintenance – that components are not simply deployed once, but they are updated and redeployed over time. Associated with this is the concept of providing continuous delivery for edge computing, building on the concepts of continuous deployment and continuous delivery as described in ISO/IEC TS 23167:2020, Clause 10.

11.5 Management of data, rights and resources

The management system needs to provide distributed data management. Elements of data are likely to exist in many nodes in the edge computing system and are also flowing between nodes. It is necessary for the data management system to be aware of the presence and location of data. The data management system has an aim to ensure the availability and integrity of the data, with a particular concern to ensure that important data is not lost. This can, for example, involve the system moving important data from nodes with limited storage capacity to secure long-term storage facilities, if new data is being continually added those nodes.

One aspect that applies to some kinds of edge computing system is the question of rights management. This can apply in situations where edge computing involves multiple different organizations and where data from one organization is transferred into the control of a second organization. In these cases, the originating organization can decide to place limits on what can be done with the data by the second organization. If this is the case, the management system needs to include rights management capabilities to ensure that the data is used only in ways that are permitted to the second organization.

The management system provides resource management. It is typical for an edge computing system to have resources with restrictions, such as limits to compute capability (processing power, memory), storage capacity, network bandwidth. The management system has to allocate resources appropriately, potentially dynamically as the demands on the system change. This includes network management, which can include the use of virtual networks as described in ISO/IEC TS 23167:2020, Clause 13.

11.6 Security and privacy management

Security and privacy (protection of PII) have associated management capabilities. The management system aims to address the security of individual components and also the security of the orchestrated sets of components. Similarly, the management system has to be aware of the presence of PII and aims to ensure that the PII is protected appropriately, wherever it is in the edge computing system. It is desirable for the management system to recognise that additional PII can in some cases be generated through the bringing together of different data elements (e.g. data from different sensors). See Clause 13 for a full discussion of security in edge computing.

12 Virtual placement

In many edge computing systems, particularly those systems based on the use of older or less capable technologies, it is typical for the placement of specific software and storage capabilities to be decided at design time, or at the latest at the time that nodes are installed into the system. In other words, the capabilities of particular nodes in such systems are fixed and do not change over time. This is particularly the case for nodes in the device tier (e.g. sensors and actuators) but can also apply to nodes in other tiers such as the edge tier.

The advent of more capable nodes with greater processing capacity (faster processors, more memory) and more storage combined with the utilization of virtualization technologies (as described in earlier clauses in this document) and service-oriented application design changes the situation significantly. Now it is possible to place software components at whatever location is most suitable for the purposes of the application and it is also possible to both update the software as desired, to run multiple software components on the same node and to change the set of software components running on a given node.

The vision of applications as sets of interacting components and datasets (the microservices model, derived from cloud computing practices), where each component can be independently deployed and scaled and where connections between components are virtualized, allows for the concept of "right placement" of components. The location of an instance of a component can be chosen to suit the needs of the application. The choice can be influenced by important factors, such as required deadlines for processing (see Clause 14), the latency of access to other components, the latency of access to data, the volume of data to be accessed and the bandwidth available over the network connections. These factors remain unchanged, what changes is the ability of the software to adapt to them in a variety of alternative ways.

Location independence follows from use of the virtualization technologies where connections between components are made over virtual network connections. There are no fixed locations. Data accesses are also achieved via data service interfaces, which effectively virtualize the location of the data. Multiple instances of a component can run at the same time and these instances can run in one location or in multiple locations spread across the network. Where the instances run is transparent to users of that component.

Using virtualization technologies such as containers, any software component can be dynamically updated as is done in cloud computing. This addresses one of the key concerns and key limitations of older edge computing systems. It is possible to evolve the capabilities of an edge computing system by updating the software on a particular node. An example could be an edge node which is performing image analysis on a video stream supplied by a camera sensor in the device layer. The image analysis can start out performing relatively simple identification of humans (say) in the scene but evolve over time to add capabilities such as gait analysis, or face recognition, all through advances in the relevant software components, dynamically updated.

In the virtualized environment, the handling of data is also fluid and mutable over time. Data might be stored locally where it is generated, or it might be moved to a location where more processing power is available to analyse the data. Data can be replicated to multiple locations, either as a whole or in part, and potentially sharded. With the ability to move processing to the data, on demand, it is also possible to use distributed data analysis techniques ("distributed queries") where analysis of a large data set does not necessarily mean that the data must be gathered in a single central location. Instead the query or analysis is created in one place and then the results are transmitted to a distributed set of nodes where the data exists and the results sent back and collated. An example implementation of this type of technique is the Gaian database[13].

Data can also be processed using a streaming technique. This is typically associated with the use of messaging systems, such as Apache Kafka[14], where data is delivered though the distributed system as a stream of messages and can be stored and processed at whatever nodes are most suitable for the task.

Virtualization techniques do not imply that all processing moves to the edge tier. "Right placement" can still indicate the need to perform some processing centrally in a large data centre. This is likely to be the case where analysis requires a very large amount of compute resources, or where the analysis involves large quantities of (often historic) data, potentially from multiple sources. It would be difficult to place processing of this nature in the edge tier, since much of the data involved would have to be accessed remotely and the edge tier might not have the necessary processing resources available.

13 Security and privacy in edge computing

13.1 General

Edge computing has associated risks and as a result needs to incorporate appropriate controls and capabilities that satisfy business expectations and applicable regulations to secure and protect information, systems and nodes. Information security is one key element that applies across all the elements of an edge computing system. Commonly, edge computing systems also process personally identifiable information (PII) and as a result involve the need to protect that PII to provide appropriate privacy.

As indicated by the ISO/IEC 30141[2], edge computing systems also usually involve more than the aspects of security and privacy. The term trustworthiness is applied to these systems, encompassing the additional aspects of safety, reliability and resilience in addition to security and privacy. These five aspects of trustworthiness interact with one another in ways that can create challenges for edge computing systems that go beyond those that apply to other types of systems.

It is also necessary to recognise that the term cybersecurity is often applied to edge computing and to IoT systems. Cybersecurity typically involves a different viewpoint than information security, concentrating on the risks that arise from constructing "systems of systems", with nodes in multiple interacting tiers using networks that can be used to mount attacks on those systems.

Some edge computing systems, particularly those involved in manufacturing and control, also involve combining traditional operational technology systems with information technology systems. This can cause difficulties for security because operational technology systems have a different approach to security than information technology systems.

13.2 Applying foundational security principles

Foundational security principles are discussed in the OWASP Security by Design Principles[23]. Edge computing systems apply foundational security principles to:

— secure information to ensure its availability, its integrity and its confidentiality;

— secure systems to ensure that they operate to design, that they cannot be hijacked, that they have no vulnerabilities, that they are available (e.g. address DDoS attacks);

— detect attacks and incidents, record and report attacks and incidents;

— ensure that nodes only communicate with other authorised nodes and that networks are appropriately protected;

— apply all appropriate data protection principles where personal data is involved, when stored or processed on a node, or when transmitted on networks between nodes;

— ensure that access to or management of nodes in the system is subject to authentication and authorization;

— adapt the security functions to the specific architecture of edge computing;

— design security functions that can be flexibly deployed and expanded;

— provision the system to continuously mitigate attacks within a certain period of time;

— provision the system to tolerate function failures within a specified range while basic functions continue to run as designed;

— ensure that the entire system can quickly recover from failure.

Security requires a comprehensive approach to all entities in the edge computing system, applying the foundational security principles during the whole lifecycle of the system, from design through implementation, allowing for maintenance and update and finally termination or retirement.

An edge system could involve spanning trust boundaries. Edge systems are often built as "systems of systems" and the ownership and control of the component systems could belong to different organizations and/or individuals. Such trust boundaries need to be addressed as part of the overall system design.

13.3 Secure nodes and devices

Given that edge computing systems are built from a networked mesh of nodes, it is important to consider the security of the nodes and especially of those nodes in the device tier and in the edge tier, often having fewer capabilities than other nodes. It is important that the nodes and devices provide mechanisms to ensure that they operate in a trusted manner at all times. This trust ideally would be consistent across both the hardware and software elements of the node or device.

Items to consider in providing secure nodes and devices include:

— hardware root of trust;

— secure cryptographic keys;

— certificate based authentication;

— software compartmentalization;

— protection for all potential attack surfaces, with multiple mitigations for threats;

— software integrity and updateability;

— data integrity and timeliness;

— detection and reporting of incidents.

Hardware based root of trust is an approach to both defend against low-level software attacks and the basis of ensuring that only authorized software can run on the node or device. One example of hardware root of trust is covered by ISO/IEC 11889-1[28].

Cryptographic keys are essential to many of the security capabilities of any node. Having those keys stored in a hardware-protected vault is important to avoid the compromise of security capabilities. Assigning the keys to the device at creation is one strategy to employ.

Authentication is necessary for the node or device itself (e.g. in proving its identity) and for capabilities running on the node. It is typical to achieve authentication using signed certificates, using protected cryptographic keys.

Software compartmentalization, providing barriers between different software components running on a node, prevents a problem (including a breach) concerning one component from spreading to other components. The barriers could be hardware imposed or hardware assisted and special provision is often applied to sensitive areas of memory, such as the storage used for cryptographic keys. Compartmentalization can also apply to storage, so that only specific software can access and use specific areas of storage. Containers and VMs generally provide for this form of software compartmentalization, not only isolating software components from each other, but also limiting what other software components they can connect with over the network.

It is typical for any particular node or device to have multiple different attack surfaces. Protection needs to be provided for each potential threat and the countermeasures provided in depth to mitigate against the effect of a breach.

Only authorized software is allowed to run on the node or device. Integrity checking can be applied to the software to ensure that it is both authorized and that it has not been subject to tampering. A desirable capability is the ability to check the validity of software remotely. At the same time, all software ideally would be updateable so that any new vulnerabilities or threats can be addressed.

Integrity of data within the node ideally would be protected and the timely availability of the data needs to be addressed, which can form another type of integrity (i.e. out-of-date data is not used for decision making).

Detection and reporting of incidents is a key requirement. When an attack occurs, it must be detected and reported so that appropriate management action can be taken. Reporting of routine activities is also desirable.

13.4 Connectivity and network security

For edge computing systems, networks connect nodes within a tier and also connect nodes in different tiers. The networks and the connectivity that takes place over them are a fundamental aspect of edge computing. Network security is an important aspect of the edge computing systems.

Networks need security controls which ensure that only authorized components communicate with each other – this is commonly done at the level of nodes. Increasingly, particularly through the use of software defined networking, controls are applied at the level of components.

Data flowing on the networks could be confidential or PII. It must not be subject to eavesdropping or to alteration. Equally, the availability of the network to transmit the data is key, especially where the timely arrival of the data is necessary for the correct operation of the edge system. Controls to prevent interference between different data flows taking place over the same physical network are another consideration.

In addition, the security design and implementation need to take the unique features of edge computing into consideration such as:

— Lightweight security functions can be deployed on IoT devices with limited hardware resources.

— Connectivity secured using a password, passcode or passkey should be unique to each node, even if the node is subjected to a factory reset.

— All keys should be stored securely.

— Each node should have a unique and tamper-proof device identifier.

13.5 Organization of security elements

The structure of edge computing systems is complex, with heterogeneous nodes spread across the tiers in many different physical locations and with highly variable security context. As a result, the organization of the security elements of an edge computing system necessarily reflects the complexity of the system itself.

Key functional elements of security of edge computing systems are illustrated in Figure 7. In Figure 7, security elements are divided into the security operational support system (OSS), which is part of the management plane as described in 11.2, and into the four areas of Application security, Data security, Network security and Node security. Functionally, these apply across all of the tiers of the edge computing system. These functional elements relate to the control plane and management plane as described in 11.2.

The traditional trust-based security model is difficult to apply to the access of a large number of heterogeneous devices. Therefore, the security model needs to be based on the minimum authorization principle.

Isolation between networks and domains is implemented on key nodes (such as gateways) to control the scope of security attacks and risks, preventing attacks on one node from spreading to the entire network.

The security monitoring functions are seamlessly embedded into the entire edge computing architecture to achieve continuous detection and response. Automation ideally would be implemented as much as possible, but manual intervention is also required at times.

Security organization covers each tier of the edge computing architecture, and different tiers might require different security features. In addition, security monitoring, security management and orchestration, ID and access management, and security policy handling are required to ensure maximum security and reliability of the entire architecture.

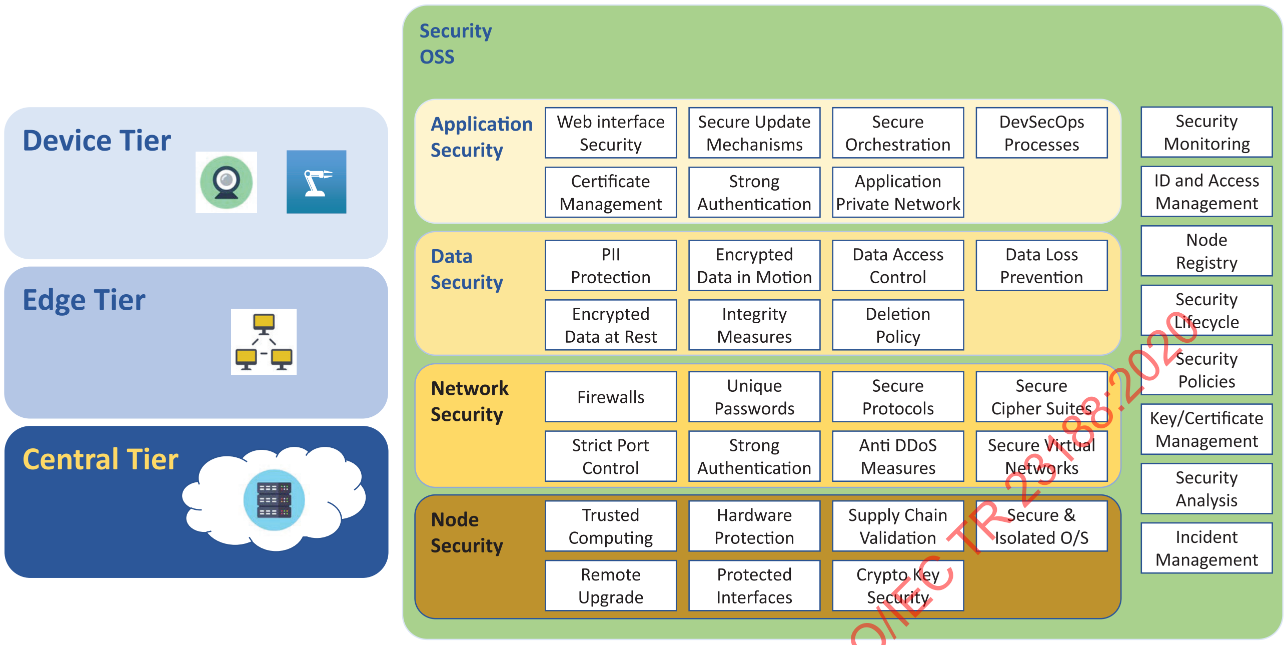

Figure 7 — Functional view of security elements

Network security:

The inherent networked basis of edge computing systems means that network security is a key element of the security of edge computing systems. This includes the use of firewalls, strict control of open ports, the use of strong authentication and unique passwords. Anti-DDoS measures are necessary, plus the use of secure protocols to address direct attacks against the networks. Dividing the networks into isolated segments linked via secure gateways is a measure that can limit the scope of an attack against one network element. The use of secure virtual networks is another important technique that can reduce the attack surface area with the potential for security at the application level.

The cipher suites used for networking need consideration. The security level of cipher suites varies and can change over time so the system needs to employ such suites flexibly and cope with changing them over time.

Data security:

Includes data encryption and/or data tokenization for both data at rest and for data in motion, data isolation and destruction, data integrity measures (anti-tampering), privacy protection (e.g. data anonymization), data access control, and data leakage prevention.

Data leakage prevention for edge computing is different from that of traditional systems because edge computing devices are usually deployed in distributed mode. Particular capabilities are required to ensure that data is not leaked even if devices are stolen.

PII protection is a significant concern, with PII minimisation, privacy-by-design and strict deletion policies in place.

Application security:

Where the application has a web interface, appropriate controls are necessary for that interface. Strong authentication is applied to all application interfaces. Traditional trust-based security model can be hard to apply to a large number of heterogeneous nodes and a multiplicity of services. Therefore, security models such as the whitelist function with minimal authorization can be useful to handle ID and access management for application components.