Journal of Big Data

METHODOLOGY Open Access

# Governance and sustainability of distributed continuum systems: a big data approach

Praveen Kumar Donta[1*], Boris Sedlak[1], Victor Casamayor Pujol[1] and Schahram Dustdar[1]

*Correspondence:
pdonta@dsg.tuwien.ac.at

[1] Distributed Systems Group, TU Wien, 1040 Vienna, Austria

**Abstract**

Distributed computing continuum systems (DCCS) make use of a vast number of computing devices to process data generated by edge devices such as the Internet of Things and sensor nodes. Besides performing computations, these devices also produce data including, for example, event logs, configuration files, network management information. When these data are analyzed, we can learn more about the devices, such as their capabilities, processing efficiency, resource usage, and failure prediction. However, these data are available in different forms and have different attributes due to the highly heterogeneous nature of DCCS. The diversity of data poses various challenges which we discuss by relating them to big data, so that we can utilize the advantages of big data analytical tools. We enumerate several existing tools that can perform the monitoring task and also summarize their characteristics. Further, we provide a general governance and sustainable architecture for DCCS, which reflects the human body's self-healing model. The proposed model has three stages: first, it analyzes system data to acquire knowledge; second, it can leverage the knowledge to monitor and predict future conditions; and third, it takes further actions to autonomously solve any issue or to alert administrators. Thus, the DCCS model is designed to minimize the system's downtime while optimizing resource usage. A small set of data is used to illustrate the monitoring and prediction of the performance of a system through Bayesian network structure learning. Finally, we discuss the limitations of the governance and sustainability model, and we provide possible solutions to overcome them and make the system more efficient.

**Keywords:** Distributed computing continuum systems, Big data analytics, Governance and sustainability, Self-healing, Representation learning, Bayesian network structure learning

## Introduction

In the past decade, a large number of resource-constrained computing devices, such as the Internet of Things (IoT), have been used in a wide range of applications. As these devices have limited resources, edge and cloud devices are added to the systems to perform larger computations. Depending on the computational effort, low-level devices (edge) are dependent on high-resource devices (cloud), which enables the edge-to-cloud continuum [1]. We refer to the entire system as Distributed Computing Continuum Systems (DCCS) [2]. DCCS are more complex than current cloud systems because of (amongst others) the heterogeneity of devices, unpredictable data sizes, geospatial distribution, varying network rates, resource federation [3].

In DCCS, tasks are divided into smaller chunks and distributed across a network of computing resources. These tasks take advantage of the combined processing power and resources of all the devices in the network [4, 5]. To coordinate their tasks efficiently, a system must manage the allocation of tasks to the available resources and the flow of data between the different devices [6]. There exist various computation and task offloading techniques in the edge-to-cloud continuum with their pitfalls [7–11]. Still, there remain several issues to solve in order to make DCCS more efficient and dynamic [12, 13]. However, the performance of DCCS not only depends on the efficiency of the computational or task offloading techniques but also the condition of the device. Specifically, this refers to whether DCCS can handle current workloads, provide reliable functionality, network connectivity, adequate energy supply, and identify other potential issues that may arise. Thus, DCCS devices need to be monitored to ensure minimal downtime, as well as to improve efficiency by utilizing resources efficiently.

Monitoring every single device in DCCS is nearly impossible, thus, there is a need for automated tools that use device data to monitor the system. Although there are some tools in the literature for monitoring IoT devices (refer Sect. "Existing solutions or tools for IoT monitoring"), they are not sufficient for the entire DCCS. Thus, novel tools and methodologies are needed to monitor the entire DCCS through efficient governance models to achieve long-term sustainability. However, the IoT tools in the literature are used to monitor, analyze data, and provide notifications or visualizations about IoT devices. To take appropriate action, these notifications require a manual analysis, which is complex and prone to errors.

DCCS requires a governance and sustainability framework that can monitor the entire system and predict upcoming failures to minimize the downtime of the system. In addition, this model minimizes human intervention and acts autonomously to resolve issues. The proposed general governance and sustainability model is based on the self-healing mechanism of the human body, due to the body's capability of diagnosing and healing issues on its own. The human body contains many cells that are used to diagnose and treat diseases, similarly, our model analyzes the data from each device to identify and predict issues. The heterogeneity of the devices in DCCS and the diversity of the data associated with them further complicate the process to analyze the data with limited resources. Within this paper, we discuss all of these challenges and suggest alternative solutions to analyze such complex data. This framework uses Representation Learning (ReL) to understand causal relationships, which helps identify or predict faults or failures of devices. Additionally, the ReL can assist in identifying the root cause of failure or faults. The identified issues are immediately visualized through Graphical User Interfaces (GUIs). Further, the appropriate administrator is informed, or issues are automatically resolved. We introduced SLOs (Service-level objectives) [14, 15] to check the reliability of our framework.

## Motivation

The self-healing power of human beings creates a dynamic equilibrium for the deprived body. With the self-healing mechanism, one can endure for a long time The human body has a brilliant self-repair system that kills cancer cells, attacks infectious agents, repairs broken proteins, keeps coronary arteries open, and prevents aging naturally [16]. Consider the simple case of how the human body heals a cut on a specific part of our body, such as a finger [17]. The human body heals the wound in four stages which include rapid hemostasis (bleeding control), appropriate inflammation, proliferation (repairing tissues), and reshaping (maturation of tissues) [18]. Each stage is associated with key molecular and cellular events as well as a multitude of secreted factors that coordinate them.

During the initial phase of hemostasis, blood cells are restricted from moving through the vessels. When a blood vessel breaks, platelets adhere to each other to seal the opening. During the inflammatory phase, the body's repair and healing cell moves to the site of the wound to protect it from infection. Bacteria, pathogens, and damaged cells are eliminated during this phase. As a result of inflammation, swelling, pain, and redness are caused by several white blood cells, enzymes, nutrients, and growth factors. A third phase of the process is proliferation, which is responsible for ensuring that newly formed blood vessels provide enough nutrients and oxygen to the granulated tissue. In the final reshaping process, the wound is closed and the collagen is reshaped as it was originally. Additionally, this phase removes dead skin cells and thickens the skin where the wound was inflicted [19, 20]. DCCS can also be modeled as a self-healing system, similar to the human body, by acting autonomously [21].

This paper models a self-governing and sustainability conceptual framework with action capabilities that are similar to how human bodies heal themselves. Governing DCCS must consider their current state and behavior. Hence, this is achieved through continuous monitoring and state forecasting based on the analysis of data collected from the devices. However, analyzing DCCS data is challenging due to the complexity associated with its data structure, and the heterogeneity of the devices. This paper presents the challenges associated with DCCS' data and provides the simplest ways to minimize downtime through governance and sustainability. The contributions of the paper are summarized in Sect. "Contribution" .

## Contribution

The major contributions of this article are summarized as follows:

1 In this paper, we proposed a general conceptual framework that can be used for developing new architectures and tools that sustainably operate for a longer period with minimal downtime through the efficient governance of DCCS. The governance and sustainability framework helps continuously monitor and predict DCCS' behavior, and heal the system if misbehavior's detected. In the end, we also summarize the challenges associated with this conceptual framework.
2 To the best of our knowledge, this is the first proposed framework in the literature to monitor the entire DCCS.
3 This paper provides an illustrative example of how Bayesian Network Structure Learning (BNSL) can be used in governance and sustainable architectures to monitor or predict DCCS's behavior.

4 Finally, we suggest possible add-ons to the governance and sustainable DCCS conceptual framework to improve the efficiency of the DCCS.

### Organization

The remaining sections of this article are arranged as follows: In section "Background", we present information about the data sources, data types, and formats in DCCS. Furthermore, we present various challenges that arise due to the data heterogeneity and differences between manufacturers. In addition, we describe how DCCS' data matches the characteristics of Big Data. Section "Problem definition" provides the problem definition and the importance of the proposed governance and sustainable conceptual framework for the near future. Section "Existing solutions or tools for IoT monitoring" discusses the most common tools available in the literature to monitor IoT networks, we examine these tools and summarize their benefits and limitations. Section "Proposed governance and sustainability framework" provides the proposed general model for the governance and sustainability of DCCS. Future research directions are covered in Sect. "Future research directions" . Finally, we conclude the paper in Sect. "Conclusion" .

## Background

In this section, we present the data associated with DCCS and its relationship to big data. In the beginning, we provide information about data sources, the varieties of data types, and formats in DCCS. Furthermore, we discuss various challenges that arise due to the heterogeneity of devices, and the diversity of device manufacturers. We briefly discuss big data and its advantages. Finally, we discuss how DCCS' data challenges match the characteristics of Big Data.

### Distributed computer continuum systems

DCCS are composed of a large number of heterogeneous computing devices, which perform large computations by coordinating their resources. The overall infrastructure of DCCS is highly scalable, flexible, and resilient to perform these computations effectively [22, 23]. During these computations, each device produces a vast amount of data, which is discussed in the following subsections.

#### *DCCS's Data*

In general, the data in DCCS will be in various forms, such as numerical, categorical, or structured data, but also raw or multimedia data. All these forms of data are distributed and processed by multiple devices across the systems. Depending on each device's capacity, either they can process complete data or a portion of the data. Delay-aware applications (e.g., industrial, medical, and transport) can partition the data among multiple devices for quick analytics.

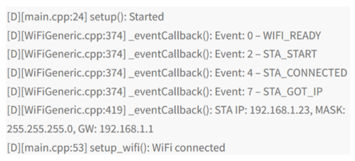
#### *Source of DCCS's data*

There is a variety of computing devices involved in DCCS, which analyses data from different sources. Besides that, each device in DCCS also maintains a variety of data in different formats, including e.g., custom log files, information about the network and devices, and location information. These data formats are elaborated as follows:

Custom Log metrics: The data available in the log files can be used for debugging, troubleshooting, monitoring, analysis, etc. Each device in DCCS maintains its own log file and notes each activity timely. While extracting the log files, one can extract the device metadata (e.g., manufacturer, model, serial number, supported data formats), specific events of the device, and its timestamp, error, or warning messages. However, multiple heterogeneous devices involved in DCCS follow different formats for their log files (e.g. CSV, JSON, or XML), thus, we can only expect the same format or attributes from similar devices in the DCCS. E.g., log files of Arduino, Raspberry pi, and a cloud machine are expected to be in the same format as shown in Fig. 1. A study on log files useful for Intrusion Detection in a system is discussed by Landauer et al. [24] and Alsaedi et al. [25].

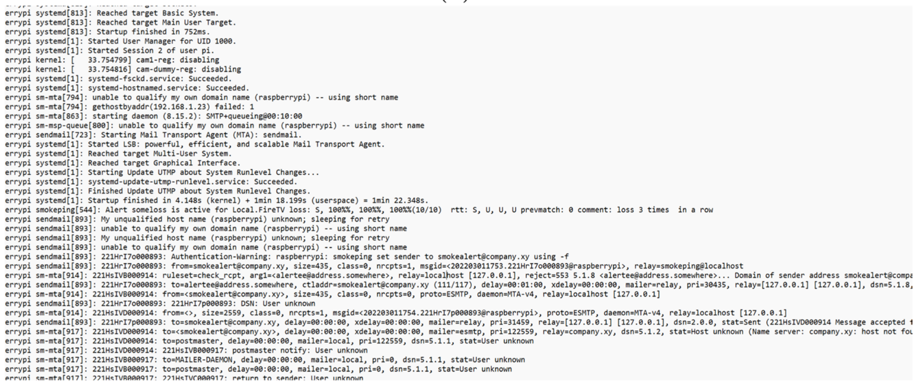
Network information: Information related to the network connection (e.g. Internet or Bluetooth) is another important data source, which provides details about the supporting communication or the network configurations. Especially in DCCS, this information is stored differently depending on the device type or its manufacturers. Some devices in DCCS have their network configuration details hard-coded into their firmware, so they cannot be changed. E.g., medical devices, such as smart inhalers or wearable fitness trackers are hard-coded to connect to Bluetooth or the Internet to transmit data or receive updates from mobile phones. Similarly, under this category comes smart home devices such as smart thermostats, security cameras, and smart plugs. In DCCS, another way to store the network configuration settings is through stored files (e.g., the static network configuration of Raspberry pi is available in '`/etc/dhcpcd.conf`') or store the network configuration details in the database of a particular device or server. E.g., the Raspberry pi dynamic and static network configuration information stored in '`/etc/wpa_supplicant/wpa_supplicant.conf`', and '`/etc/dhcpcd.conf`', respectively, and they are depicted in Fig. 2.

Hardware information: Most of the devices participating in DCCS are resource constrained. There are limitations on their hardware in terms of microcontrollers, battery capacities, Radio Frequency (RF), and ports. The hardware configuration information is stored in a file in each device, e.g., Raspberry pi maintains all its components information using a file located at the path '`/boot/config.txt`.' This file contains information related to the



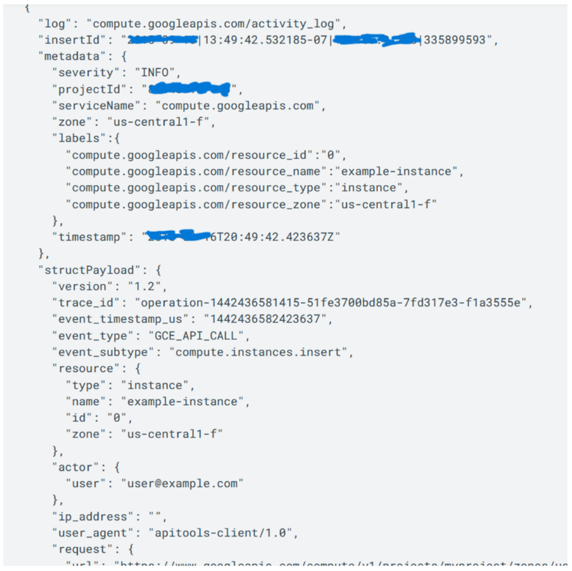
```
[D][main.cpp:24] setup(): Started
[D][WiFiGeneric.cpp:374] _eventCallback(): Event: 0 – WIFI_READY
[D][WiFiGeneric.cpp:374] _eventCallback(): Event: 2 – STA_START
[D][WiFiGeneric.cpp:374] _eventCallback(): Event: 4 – STA_CONNECTED
[D][WiFiGeneric.cpp:374] _eventCallback(): Event: 7 – STA_GOT_IP
[D][WiFiGeneric.cpp:419] _eventCallback(): STA IP: 192.168.1.23, MASK:
255.255.255.0, GW: 192.168.1.1
[D][main.cpp:53] setup_wifi(): WiFi connected
```

(a)



```
errypi systemd[813]: Reached target Basic System.
errypi systemd[813]: Reached target Main User Target.
errypi systemd[813]: Startup finished in 752ms.
errypi systemd[1]: Started User Manager for UID 1000.
errypi systemd[1]: Started Session 2 of user pi.
errypi kernel: [   33.754799] cam1-reg: disabling
errypi kernel: [   33.754816] cam-dummy-reg: disabling
errypi systemd[1]: systemd-fsckd.service: Succeeded.
errypi systemd[1]: systemd-hostnamed.service: Succeeded.
errypi sm-mta[794]: unable to qualify my own domain name (raspberrypi) -- using short name
errypi sm-mta[794]: gethostbyaddr(192.168.1.23) failed: 1
errypi sm-mta[863]: starting daemon (8.15.2): SMTP+queueing@00:10:00
errypi sm-msp-queue[800]: unable to qualify my own domain name (raspberrypi) -- using short name
errypi sendmail[723]: Starting Mail Transport Agent (MTA): sendmail.
errypi systemd[1]: Started LSB: powerful, efficient, and scalable Mail Transport Agent.
errypi systemd[1]: Reached target Multi-User System.
errypi systemd[1]: Reached target Graphical Interface.
errypi systemd[1]: Starting Update UTMP about System Runlevel Changes...
errypi systemd[1]: systemd-update-utmp-runlevel.service: Succeeded.
errypi systemd[1]: Finished Update UTMP about System Runlevel Changes.
errypi systemd[1]: Startup finished in 4.148s (kernel) + 1min 18.199s (userspace) = 1min 22.348s.
errypi smokeping[544]: Alert someloss is active for Local.FireTV loss: S, 100%%, 100%%, 100%%(10/10)  rtt: S, U, U, U prevmatch: 0 comment: loss 3 times  in a row
errypi sendmail[893]: My unqualified host name (raspberrypi) unknown; sleeping for retry
errypi sendmail[893]: unable to qualify my own domain name (raspberrypi) -- using short name
errypi sendmail[893]: My unqualified host name (raspberrypi) unknown; sleeping for retry
errypi sendmail[893]: unable to qualify my own domain name (raspberrypi) -- using short name
errypi sendmail[893]: 221HrI7o000893: Authentication-Warning: raspberrypi: smokeping set sender to smokealert@company.xy using -f
errypi sendmail[893]: 221HrI7o000893: from=smokealert@company.xy, size=435, class=0, nrcpts=1, msgid=<202203011753.221HrI7o000893@raspberrypi>, relay=smokeping@localhost
errypi sm-mta[914]: 221HsIVB000914: ruleset=check_rcpt, arg1=<alertee@address.somewhere>, relay=localhost [127.0.0.1], reject=553 5.1.8 <alertee@address.somewhere>... Domain of sender address smokealert@compa
errypi sendmail[893]: 221HrI7o000893: to=alertee@address.somewhere, ctladdr=smokealert@company.xy (111/117), delay=00:01:00, xdelay=00:00:00, mailer=relay, pri=30435, relay=[127.0.0.1] [127.0.0.1], dsn=5.1.8,
errypi sm-mta[914]: 221HsIVB000914: from=<smokealert@company.xy>, size=435, class=0, nrcpts=0, proto=ESMTP, daemon=MTA-v4, relay=localhost [127.0.0.1]
errypi sendmail[893]: 221HrI7o000893: 221HrI7p000893: DSN: User unknown
errypi sm-mta[914]: 221HsIVD000914: from=<>, size=2559, class=0, nrcpts=1, msgid=<202203011754.221HrI7p000893@raspberrypi>, proto=ESMTP, daemon=MTA-v4, relay=localhost [127.0.0.1]
errypi sendmail[893]: 221HrI7p000893: to=smokealert@company.xy, delay=00:00:00, xdelay=00:00:00, mailer=relay, pri=31459, relay=[127.0.0.1] [127.0.0.1], dsn=2.0.0, stat=Sent (221HsIVD000914 Message accepted f
errypi sm-mta[917]: 221HsIVD000914: to=<smokealert@company.xy>, delay=00:00:00, xdelay=00:00:00, mailer=esmtp, pri=122559, relay=company.xy, dsn=5.1.2, stat=Host unknown (Name server: company.xy: host not fou
errypi sm-mta[917]: 221HsIVD000914: to=postmaster, delay=00:00:00, mailer=local, pri=122559, dsn=5.1.1, stat=User unknown
errypi sm-mta[917]: 221HsIVD000914: 221HsIVB000917: postmaster notify: User unknown
errypi sm-mta[917]: 221HsIVB000917: to=MAILER-DAEMON, delay=00:00:00, mailer=local, pri=0, dsn=5.1.1, stat=User unknown
errypi sm-mta[917]: 221HsIVB000917: to=postmaster, delay=00:00:00, mailer=local, pri=0, dsn=5.1.1, stat=User unknown
errypi sm-mta[917]: 221HsIVB000917: 221HsIVC000917: return to sender: User unknown
```

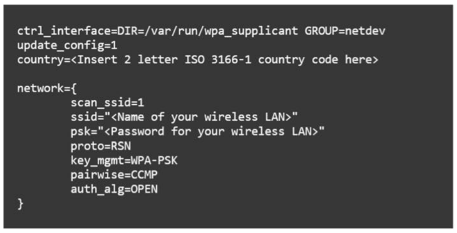
(b)



```
{
  "log": "compute.googleapis.com/activity_log",
  "insertId": "[redacted]|13:49:42.532185-07|[redacted]|335899593",
  "metadata": {
    "severity": "INFO",
    "projectId": "[redacted]",
    "serviceName": "compute.googleapis.com",
    "zone": "us-central1-f",
    "labels":{
      "compute.googleapis.com/resource_id":"0",
      "compute.googleapis.com/resource_name":"example-instance",
      "compute.googleapis.com/resource_type":"instance",
      "compute.googleapis.com/resource_zone":"us-central1-f"
    },
    "timestamp": "[redacted]6T20:49:42.423637Z"
  },
  "structPayload": {
    "version": "1.2",
    "trace_id": "operation-1442436581415-51fe3700bd85a-7fd317e3-f1a3555e",
    "event_timestamp_us": "1442436582423637",
    "event_type": "GCE_API_CALL",
    "event_subtype": "compute.instances.insert",
    "resource": {
      "type": "instance",
      "name": "example-instance",
      "id": "0",
      "zone": "us-central1-f"
    },
    "actor": {
      "user": "user@example.com"
    },
    "ip_address": "",
    "user_agent": "apitools-client/1.0",
    "request": {
      "url": "https://www.googleapis.com/compute/v1/projects/myproject/zones/us
```

(c)

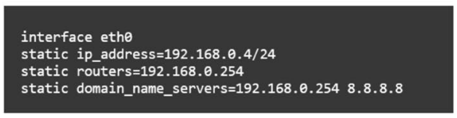
**Fig. 1** Example for System log files of different devices in DCCS **a** Arduino **b** Raspberry Pi (**c**) Cloud server



```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=<Insert 2 letter ISO 3166-1 country code here>

network={
        scan_ssid=1
        ssid="<Name of your wireless LAN>"
        psk="<Password for your wireless LAN>"
        proto=RSN
        key_mgmt=WPA-PSK
        pairwise=CCMP
        auth_alg=OPEN
}
```

(a)



```
interface eth0
static ip_address=192.168.0.4/24
static routers=192.168.0.254
static domain_name_servers=192.168.0.254 8.8.8.8
```

(b)

**Fig. 2** Example for Network configuration information of Raspberry Pi **a** Dynamic **b** Static

hardware, such as input/output settings, multimedia (audio/video) support, HDMI display setting, supporting resolutions, the bus and its speed, and GPU information (if any). Through this file, one can adjust the settings as necessary for the need of an application. This information helps in deciding whether the device can participate in the computation or needs to transfer the data to a different device with higher resource availability. It is essential to monitor these settings as inappropriate configuration values can cause the device to become unstable. The information about energy availability and usage is an additional important metric for the DCCS devices, because most of them are operated through external energy sources. These values are grasped using separate power monitoring meters or software-based solutions.

Device location: The locations of all devices contained in the DCCS are essential because they determine how well the devices can communicate with other devices. A device location is considered in two ways: Physical location and logical location. The device's physical location is represented with geographic coordinates, which are notated using longitude and latitude. This data can be fetched from the metadata or the server's database. In the case of mobile devices (e.g., autonomous vehicles), a GPS tracker is attached to track their physical locations. The logical location indicates the device's location in the network, such as topological order or IP address. The logical location of a device is stored in the '`/etc/hosts`' (static hosts), which helps to identify whether the device is active or not. The pointing direction of a device is also crucial for some applications, such as camera devices for surveillance.

### *Challenges of DCCS's data*

The data of DCCS are in different formats and they lead to further complications in analyzing and extracting knowledge from it. These challenges are categorized as follows:

High non-linearity among attributes: Due to the heterogeneity of the devices in the DCCS, there are huge diversities in their attributes. Even though some devices are homogeneous, but the data in DCCS is diversified in terms of usage patterns, performance metrics, etc., over time. These attributes cause non-linearity, which leads to complexity in identifying the relationship between the dependent and independent attributes. For example, throughput and cost of Virtual Machines (VMs) in an elasticity-enabled cloud are non-linear [26]. However, the

| | |
|---|---|
| | workload and cost of VMs are linear when considering the pay-as-use principle. |
| Sparsity: | The sparsity in DCCS's data indicates the unavailability of specific data, missing any details or the data contains only zeros. It can also be defined as the data in which only a small fraction of it is useful; extracting useful information out of sparse data is a challenging issue. It is more complicated when the datasets are non-linear, e.g., Fig. 1 depicts the same devices participating in DCCS tasks having different forms of logs whereas some of the attributed information from Fig. 1a, b, and c are not common. These sparse data make it difficult to perform analytics. Yuejie et al. in [27] provide an approach using atomic norm minimization to harness sparsity over the continuum. Federico et al. [28] provide a detailed study on sparse and dense data in IoT. |
| Structure breaking: | The structure of all the systems data in DCCS is not similar. It is a combination of databases, file systems (e.g. configuration or setup files), semi-structured data, and sparse data. An advantage of using structure breaking is that it is easy to modify or analyze the data. Since we are working with the whole system, it is necessary to preserve the structure of systems data. Structure-preserving models are characterized by preserving explicit properties of their counterparts in their discretizations. But, preserving the structure of DCCS' data is challenging. |
| Explanatory factors across the systems: | Since the DCCS' data contains non-linearity, sparsity, and structure breaking, it also contains explanatory factors across the systems. These explanatory factors help to minimize the number of computations required to make the decisions in the systems. For instance, if the response from a device is not received for a while, our analytics can guess either the device is down due to lack of energy or disconnected from the Internet. In the same case, the device is responding for some time and not responding at other times, this indicates that the networking of the particular device is faulty. In case, the |

device does not respond completely for a longer period, this may be due to the drain of complete energy. The explanatory factors are of different types:

| | |
|---|---|
| Simple explanatory factors: | These explanatory factors show a relationship between a maximum of two dependent variables. For example, consider *Packet_loss* and *Throughput* as variables. Decreasing the *Packet_loss* in the system increases the *Throughput*. Simple explanatory factors help to break large and complex systems into simpler ones. |
| Multiple explanatory factors: | These kinds of variables have a relationship between multiple dependent attributes. The effect of one variable affects multiple other variables. Consider, e.g., variables such as *Packet_loss*, *Buffer_usage*, and *Delay*: the overflow of *Buffer* affects multiple other attributes such as increases *Packet_loss* and *Delay*. Hence, changing a variable directly affects multiple other variables. |
| Hierarchical explanatory factors: | These kinds of factors are dependent hierarchically. Which indicated the effect on variable *x* causes an effect on *y*, the effect on variable *y* causes an effect on *z*, and so on. For e.g., sharing the same portion of data to multiple devices in DCCS cause additional wastage of energy and computational resources. Additional resource utilization cause *Delay* to compute the data available for analytics in the queue. Longer waits in the queue may cause buffer overflows. *Buffer* overflow causes increased *Packet_loss*, and increased *Packet_loss* causes lowing *Throughout*, and so on. |

To overcome all the above challenges, an efficient analysis model is required to extract useful information for further decision-making, and Big Data Analysis is a great solution for such a purpose.

## Big data and its characteristics

A vast number of machines and humans interacting with them continuously generate structured, semi-structured, or unstructured data. These massive amounts of data cannot be handled through conventional storage or processing approaches[29]. Big data is a term used to describe an immense amount of data that is too complex and too large to be processed and analyzed using traditional data processing tools [30]. Big data has become increasingly important in recent years because it allows organizations to gain insights from this data. By analyzing large volumes of data, organizations can uncover hidden patterns, correlations, and trends that can help them make better decisions and improve their operations [31]. It is generated by a variety of sources, such as sensors, social media, agriculture [32] and transactions, and it is typically stored in distributed systems such as Hadoop or NoSQL databases. Because of its size and complexity, big data requires specialized tools and techniques to process and analyze the data, such as machine learning (ML) and distributed computing. Big data is often characterized by the 5Vs: volume, velocity, variety, value, and veracity [33, 34], which are summarized as follows:

| | |
|---|---|
| Volume: | The basic characteristic of Big Data is that large amounts of data are generated or made available or stored for computations. |
| Velocity: | The velocity is defined as the speed of the streaming data [35]. As DCCS' data is streaming and changing every unit of time, this metric refers to the speed at which the data is generating or changing. This metric also refers to the movement of the data from one device to another. |
| Variety: | It is termed as different types of data that are available or generated for computations. This means that the data is a combination of all structured, semi-structured, and unstructured data which can be stored or meant for computation. |
| Value: | This characteristic is related to a specific application. This refers to a potential value (insights) that can be achieved through analysis, and it is useful to make efficient decisions depending on the applications. |
| Veracity: | This refers to the consistency, reliability, quality, or accuracy of the data. Sometimes, there is missing or inconsistent information in the given input data, which cannot provide deeper insight after analysis. In short, this characteristic defines the level of trust in the DCCS' data. |

Characteristics such as *Value* and *Veracity* help to gain insights and achieve the quality of the big data.

### *How does DCCS's data meet big data characteristics?*

When we compare the challenges of DCCS's data discussed in subsection  such as non-linear, sparsity, explanatory factors, and structure breaking, we can claim that it satisfies the 5Vs of big data.

Multiple and vast numbers of devices are involved in DCCS, and each device is associated with its data. When data from all devices are combined, it will become large *volume* of data [22, 36, 37]. The challenge of *structure breaking* in DCCS's data (which are in the form of file (unstructured/semi-structured), database (structured), etc.) implies the nature of *Variety* characteristic of big data. The Explanatory challenge factors across the systems refer to the dependencies or linearity among multiple attributes in the system's data. This characteristic of DCCS's data allows us to gain insights, achieve meaningful, reliable, or trustable data. So, this challenge of DCCS's data satisfies the *Value* and *Veracity* characteristics of big data. Since the DCCS are dynamic and operated in real-time, most of the metrics or data associated with each device change frequently. This dynamically changing data is called as streaming data which is also to be analyzed. In contrast, the streaming data must meet the characteristic of *Velocity* of the big data.

So, we can claim that the data associated with the DCCS' devices meet all the Big data characteristics.

## Problem definition

DCCS performance depends on resource efficiency, but also on maintaining device health. For example, each device is properly connected to the network, buffer usage is appropriate, functionalities are reliable, able to handle assigned workloads efficiently, etc. These conditions are continuously monitored and ensure minimal downtime to achieve maximum efficiency out of DCCS. These are more complex than a current Cloud-based systems because of the vast amount of heterogeneous devices, data sizes, geospatial locations, communication channels, etc. Monitoring each system manually is nearly impossible. Also, to the best of our knowledge, there is no system or tool available in the literature to fulfill this task.

In this context, we propose a conceptual framework to monitor the entire DCCS autonomously. Our framework addresses efficient decision making, and provides self-healing strategies to achieve long-term system sustainability. It is able to analyze the data associated with each device and make decisions in terms of the health condition of the device. Through a self-healing mechanism similar to the human body, the fault or failure in the system is autonomously healed. But, there are several challenges associated with DCCS's data, which are summarized in Subsection . However, we have alternative solutions to mitigate these challenges. By considering all these challenges, DCCS are continuously monitored and predictive analytics are provided through the proposed governance and sustainable conceptual framework. There are several advantages of using the proposed governance and sustainable conceptual framework are discussed as follows:

| | |
|---|---|
| Early detection of fault or failures: | The continuous monitoring of the entire DCCS help to identify fault or failures before they might occur in the system. |
| Performance enhancement: | The monitoring framework can help to identify bottlenecks in DCCS, so it is easy to rectify them to improve the performance of the system. |
| Scalability: | The Framework allows to handle a large amount of data and provides insights into complex data. |
| Efficient resource utilization: | Through predictive analytics, it is possible to optimize resource utilization such as bandwidth, buffer, and computing capabilities. |

## Existing solutions or tools for IoT monitoring

There are several tools used in the literature to monitor IoT devices. However, each tool has its own limitations and benefits. Some of the currently used popular tools are summarized in Table 1. In Table 1, we summarize prominent features including cloud-based, visualization, scalability, security, complexity and open source or commercial. As we noticed, except Particle all other tools listed in this section are cloud-based.

*Visualization* identifies whether the tool can visualize the analyzed result to identify and monitor faults. *Scalability* shows whether the tool can provide services if any newly added devices included in to the existing IoT system. The *security* feature indicates whether the tool provides secure data analytics or not. *Complexity* means how much computational resource or time is needed to analyze monitoring data. Here, *high* indicates the tool needs more computational capabilities to analyze, and vice versa. *Open source* refers to whether the tool is available for free use or commercial. Following is a further discussion of each tool.

| | |
|---|---|
| Domotz: | This is a cloud-based remote network monitoring tool to sustain the network as long as possible by troubleshooting issues. Domotz can monitor the performance of large networks and their configurations, and if any malicious behavior is identified alert them to the administrators. |
| Datadog: | This tool collects and analyzes business data along with device data to identify the performance and health of the device. Datadog supports multiple cloud platforms (e.g. Google, Azure, AWS), and is also available in multiple languages such as Python, PHP,.NET, Ruby, etc. |
| Particle: | It is an edge-to-cloud IoT platform that monitors the device's health remotely and mitigates any rogue emissions identified in an event log file. This tool can identify the root cause of data leaks, if any, and notify the administrators immediately. |
| MetricFire: | This is an open-access tool for monitoring IoT devices, which are located either locally or in a remote cloud system. This tool uses Graphite to acquire the data and |

**Table 1** List of available centralized/remote IoT device monitoring tools

| Tools | Cloud-based | Visualization | Scalability | Security | Complexity | Open source |
|---|---|---|---|---|---|---|
| Domotz [38] | ✓ | ✓ | × | ✓ | High | × |
| Datadog [39] | ✓ | ✓ | ✓ | × | High | × |
| Particle [40] | × | × | ✓ | × | Medium | ✓ |
| MetricFire [41] | ✓ | ✓ | × | × | Low | ✓ |
| ThingWorx [42] | ✓ | ✓ | × | × | Medium | × |
| Splunk [43] | ✓ | × | ✓ | × | High | × |
| Senseye PdM [44] | ✓ | × | × | ✓ | Medium | × |
| SkySpark [45] | ✓ | ✓ | × | × | Low | ✓ |
| Oracle IoT [46] | ✓ | × | × | × | High | × |
| AWS IoT Monitoring [47] | ✓ | ✓ | × | × | Medium | ✓ |
| Salesforce IoT Cloud [48] | ✓ | × | ✓ | × | Low | × |
| Azure IoT Suite [49] | ✓ | ✓ | × | × | Medium | ✓ |
| IBM Watson IoT [50] | ✓ | × | ✓ | × | High | × |
| TeamViewer IoT [51] | ✓ | ✓ | × | ✓ | High | × |

| | |
|---|---|
| | visualize the performance of devices. MetricFire can work in any network size of edge devices and provides user-friendly visualization to identify the issues quickly by the administrator. |
| ThingWorx: | This is a cloud-based remote monitoring tool that is designed for Industrial IoT for predictive maintenance. ThingWorx can monitor irrespective of the number of devices in the network. |
| Splunk: | This tool acquires device data and extracts insights to identify the device functionalities and keep informing the administrator of performance gaps to enable effective troubleshooting. Splunk analytics provides customized metrics that can be visualized to the user through different tools. |
| Senseye PdM: | This is a cloud-based remote monitoring tool for Industrial IoT devices; it provides predictive maintenance through AI/ML-based algorithms. Senseye PdM is efficient in producing valid predictions which include the lifespan of a machine. This tool can support Microsoft Azure cloud. However, this tool is not efficient for a small network with a limited number of IoT devices. |
| SkySpark: | This is an open analytical platform to perform the analytics of the data collected from the sensors, extract insights, and visualizes these insights to the user. |
| Oracle IoT: | This tool is designed for industrial IoT device monitoring based on data collected from devices. Oracle IoT can be connected to Oracle apps to visualize the devices' metrics. |
| AWS IoT Device Monitoring: | This tool can monitor irrespective of the number of devices in the network and support multiple IoT application protocols including HTTP, MQTT, and WebSocket (more about these protocols can be found in *Donta et al.* [52]). This tool can run centrally in the AWS cloud platform, and provide secure networking. |
| Salesforce IoT Cloud: | This tool acquires business and device data through RESTful and API for further analysis. |
| Microsoft Azure IoT Suite: | This is a cloud-based intelligent remote monitoring tool that diagnoses edge devices, provides customized metrics, and visualizes them through various tools including digital twins. This tool can support irrespective of the size of the edge devices. |
| IBM Watson IoT: | This is one of the IoT device health monitoring platforms. IBM Watson IoT monitors the functionalities, connectivity, and availability of edge devices through AI-based data analytics. This tool works intelligently and |

| | |
|---|---|
| | supports cognitive computing approaches to enhance decision-making efficiency. This tool supports the IBM cloud. |
| TeamViewer IoT: | This tool acquires data from industrial IoT devices and monitors the devices remotely and ensures the security of IoT devices. |

All the tools discussed in this section monitor only a specific part (IoT devices), but not the entire structure of DCCS. Moreover, most of these tools monitor the software or hardware of a specific IoT device, but not their connectivity and network issues. To the best of our knowledge, there are no tools available in the literature to monitor the entire DCCS. Also, no self-healing mechanisms were implemented for tools summarized in Table 1. Thus, there is a need to have an emerging tool to fill this gap.

## Proposed governance and sustainability framework

In this section, we define a general framework for monitoring and predicting system performance and its condition continuously. We also define necessary actions to take place automatically or manually to maintain a minimal downtime of DCCS. The general framework for the proposed DCCS governance and sustainable model is presented in Fig. 3. In this model, there are multiple phases including data acquisition from devices, representation, knowledge extraction, monitoring DCCS, forecasting troubles, notifying or visualizing troubles to administrators, and taking corrective actions on their own or through administrators. In this section, we also provide an illustrative example for a better understanding of the proposed governance and sustainable architecture through BNSL. Lastly, this section concludes with a discussion of challenges imposed by DCCS sustainability and governance process.

### Data

In the current digital age, data plays a crucial role in several fields e.g., Business, Healthcare, Industry automation, Education, and DCCS data is not an exception. DCCS applications use business data analytics, but the data associated with the devices is usually ignored. There is a lot of data available or generated by each device, which will be discussed in Subsection  and it comes in different formats. These data can be used as input to the governance and sustainability model to perform efficient monitoring

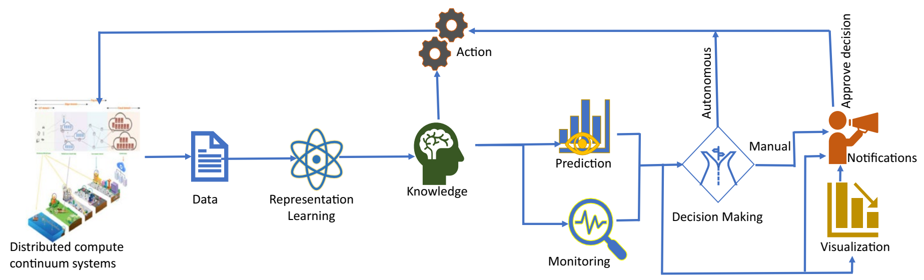


**Fig. 3** The proposed governance and sustainability conceptual framework for DCCS

and predictions about the DCCS condition. Since DCCS data is in different formats (unstructured), various methods are mentioned in the literature to convert it to structured data such as data modeling, text mining [53], data wrangling [54, 55], natural language processing, etc. Evaluation metrics such as detection accuracy or false alarm rates require ground truth tables that assign labels unambiguously to all events. Moreover, big data analytics tools can perform them without having to convert data from unstructured to structured form [56, 57].

Because DCCS data poses several challenges (subsection ), it leads to regular issues in ML datasets such as missing values, high dimensionality, label imbalance, lack of generalization, and noisy data. Several methods have been proposed in the literature to mitigate these challenges, including feature engineering, data preprocessing, data augmentation, and dimensionality reduction [58]. Nevertheless, addressing these challenges requires additional computing resources, which is a complex task for constrained edge devices in DCCS (since we are distributing the loads among various parts of the systems). In addition, it requires knowledge of data that is being converted from unstructured to structured. Here, representation learning plays a significant role in extracting the underlying features from unstructured and non-linear data, which helps to mitigate the challenges discussed in subsection .

## Learning representations from data

Many preprocessing approaches have been proposed in the literature for mitigating missing, noisy values, outliers, and inconsistent details within the given data. They are efficient at performing these tasks but they do not ease the extraction of knowledge from them. So, learning data representation makes the knowledge extraction process easier and helps to build efficient predictors and classifiers. The ReL strategies consider a vast amount of data and perform multiple transformations to extract useful information from it [59]. They also represent it in a way that is easily accessible, such as through graphs. There are several benefits of using representation learning including scalability, structure preservation, information preservation, data sparsity, and easy-to-identify explanatory factors among the data [60, 61]. Due to these reasons, learning representation approaches are used in several applications such as Natural Language Processing (NLP), speech recognition, and signal processing [62]. To simplify the process of knowledge extraction, we should also consider learning representation for resource-constrained devices, such as DCCS [2].

DCCS data contain several complexities which are discussed in Sect. "Challenges of DCCS's data", while representation learning can handle all of those complexities. In addition, ReL can minimize manual preprocessing or data modeling time. It reduces the dimensionality of data and compares large amounts of data to useful features. ReL can remove inconsistencies, noisy or outlier values, and recover missing values from the data during this process. Because the dimension is lowered, learning algorithms require less effort to analyze and produce knowledge. On the other hand, the best useful features help to improve the accuracy of the results. There are several ReL algorithms in the literature, and the most popular are Graph Representation Learning (GRL), Contrastive Representation Learning (CRL), Bayesian Network Structure Learning (BNSL), Matrix factorization, Random walk learning (RWL), Deep walk, etc [23]. We present an illustrative example to better understand how ReL (through BNSL) can generate structures, which further supports knowledge discovery.

### *Illustrative example*

We provide an illustrative example to better understand a ReL of the proposed governance and sustainable model. In this example, we consider simple data and monitor the performance of DCCS using a ReL approach called Bayesian network structure learning. We thought of simplified values of five devices (device$_1$ to device$_5$) and four performance metrics (Throughput (*TH*), Buffer usage (*BU*), Packet loss (*PL*), and Delay (*DE*)), as shown in Eq. (1). Using these values, we learn the representation of the metrics and decide how they are causally related to each other. The values in Eq. (1) are represented with binary values, in which the value *one (1)* indicates the device is performing well (i.e., meets the threshold), whereas *zero (0)* indicates the device does not meet the threshold. e.g., the throughput of *device*$_4$ is poor, and the remaining devices are performing well. Similarly, the *BU* of *device*$_3$ is under the threshold when compared with the remaining devices, and meets the threshold for the remaining three metrics, throughput, packet loss, and delay.

$$\begin{array}{c c} & \begin{array}{cccc} Th & BU & PL & De \end{array} \\ \begin{array}{c} device_1 \\ device_2 \\ device_3 \\ device_4 \\ device_5 \end{array} & \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \end{array} \tag{1}$$

We consider the BNSL approach and build the relations among the metrics according to the data given in Eq. (1). The BNSL provides a learned structure in the form of a knowledge graph specifically a Direct Acyclic Graph (DAG), helping to identify the relation between the direct and indirect correlations [63, 64]. Since BNSL can be used in constrain-based or score-based, in this paper we consider a score-based approach [65]. The score metric for a structure *G* and data *D* can be generally defined as shown in Eq. (2)

$$Score(G, D) = LL(G, D) \pm Bias \tag{2}$$

where the first term $LL(G, D)$ indicates the log-likelihood of the data under DAG structure *G*. The second term is used for regularization, i.e., *Bias Regularization*. One of the popular Bias regularization functions is $-\phi(|D|) \times ||G||$, which is known as neutralized function.

Here the first term, i.e. $\phi(|D|)$ is calculated using Eq. (3)

$$\phi(|D|) = \frac{log(|D|)}{2} \tag{3}$$

where $|D|$ indicates the number of data points. The $\phi(|D|)$ is known as the Bayesian Information Criterion (BIC). With the BIC, the influence of model complexity decreases as $|D|$ grows, allowing the log-likelihood term to eventually dominate the score.

The second term $||G||$ indicates the sum of parameters or components available in $G$, as shown in Eq. (4).

$$||G|| = \sum(|E| + |V|) \tag{4}$$

where $|E|$ indicates total number of edges, and $|V|$ indicates total number of vertices in $G$.

Initially, we build a DAG using chow-liu algorithm [66] and identify the optimal BNSL using the scope function shown in Eq. (2). The Initial complete Graph generated using the given data shown in Eq. (1) is represented as shown in Fig. 4a. Construction of score-based DAG for Fig. 4a using chow-liu algorithm use the following steps:

1 Compute Mutual Information (MI) of each possible edge using Eq.(5)

$$MI(X,Y) = \sum_{x\in X}\sum_{y\in Y} P(x,y) log \frac{P(x,y)}{P(x)P(y)} \tag{5}$$

where $P(x)$, and $P(x, y)$ are calculated as shown in Equation (6) and Equation (7), respectively.

$$P(x) = \frac{count(x)}{\text{number of data points}} \tag{6}$$

$$P(x,y) = \frac{count(x,y)}{\text{number of data points}} \tag{7}$$

The resultant MI for each edge such as $(\overline{TH,BU}) = 0.0729$, $(\overline{TH,PL}) = 0.322$, $(\overline{TH,De}) = 0.171$, $(\overline{BU,PL}) = 0.322$, $(\overline{PL,DE}) = 0.002$, and $(\overline{BU,DE}) = 0.322$. Figure 4b shows the graphical representation for weighted MI values of each edge in graph $G$.

2 Next, determining Maximum Spanning Tree (MST) using the MI weights for the Fig. 4b, whereas this work considers Kruskal's algorithms [67] for this job. Initially, we consider a Tree ($\mathcal{T}$) without any edges as shown in Fig. 5a. Next, we choose the maximum MI value from the weighted graph (Fig. 4b), and include it to ($\mathcal{T}$), if it does not form any cycle. In this case, we identify three MIs having similar values and

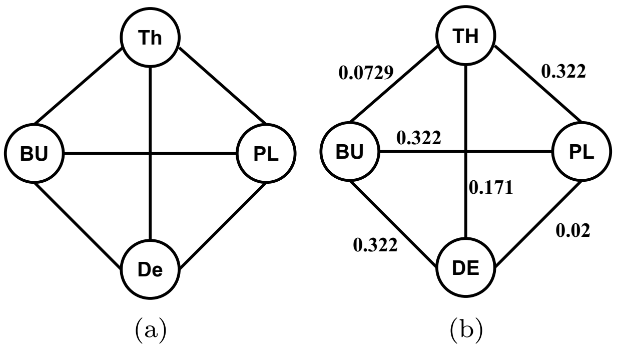


**Fig. 4** Bayesian network Structure Learning (a) Initial Complete Graph for input data shown in Eq. (1) (b) Mutual information of each possible edge

tie the break using the order of the attributes in the dataset. Further, we consider the next highest weight MI which is $(\overline{PL, BU})$, as shown in Fig. 4c. We iterate the process again and choose the next highest edge i.e. $(\overline{BU, DE})$, as shown in Fig. 4d. We repeat this process until the number of edges in $\mathcal{T}$ is one less than the number of nodes (*i.e.*, $|V| = |N| - 1$), where $N$ and $V$ are the number of nodes and edges in $\mathcal{T}$, respectively.

3 Once the $|N| - 1$ edges included to $\mathcal{T}$, it is necessary to *identify the root node* among them. Instead of randomly choosing the root node, the chow-liu algorithm uses *Log-likelihood* function to estimate the value of choosing each node as a root and decide the best one according to the maximum value returned by it. The log-likelihood function is shown in Eq. (8).

$$LL(D) = -N \times \sum_{i}^{N} \varrho(x_i|Parent(x_i)) \tag{8}$$

where $N$ is the number of data entries, $D$ denotes the DAG, and $\varrho$ is denoted as entropy and it is computed using Eq. (9)

$$\varrho(x_i|Parent(x_i)) = \sum Pr(x_i|Parent(x_i)) \times \log(Pr(x_i|Parent(x_i))) \tag{9}$$

We consider each node as a route and estimate the $LL(D)$ when considering them as a route and also assign the directions for the MST. The $LL(D)$ when considering the $TH$ as a root is $\approx (-5.033)$, and the directions are assigned as shown in Fig. 6a. Next, we change the root node to $BU$, and estimate the $LL(D)$, and it results in a value similar to $TH$ i.e., $\approx (-5.033)$. The directions for each edge are decided as shown in Fig. 6b, when $BU$ is the root. Further, we estimate the $LL(D)$ by choosing $PL$ as a root, the resultant value is $\approx (-5.6925)$, and the directions of edges are represented in Fig. 6c. Finally, the node $DE$ is considered the root, the $LL(D)$ returned a value of $\approx (-7.359)$, and the direction of $D$ is decided as shown in Fig. 6d. From this, we notice a similar value for the nodes $TH$ and $BU$, whereas we can consider any DAG for further BNSL.

We consider node $BU$ as the root and continue the construction of BNSL using it. For simplicity of presentation, we adjust the DAG without changing the direction of the

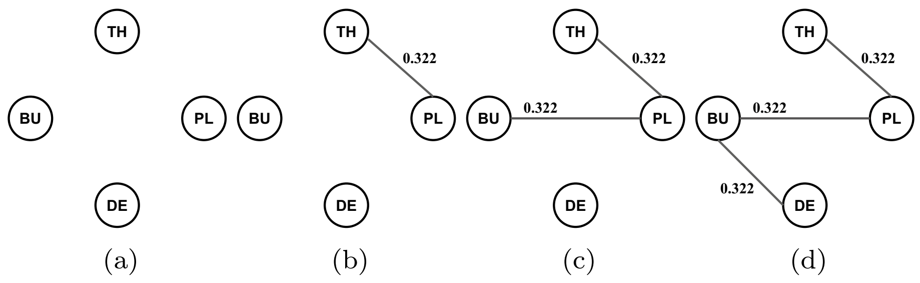


**Fig. 5** Maximum spanning tree construction using Kruskal's algorithms (a) Initial Tree with empty edges (b) Considering first maximum MI weight which is $(\overline{TH, PL})$ (c) Considering second maximum MI weight which is $(\overline{PL, BU})$ (d) Considering next maximum MI weight which is $(\overline{BU, DE})$

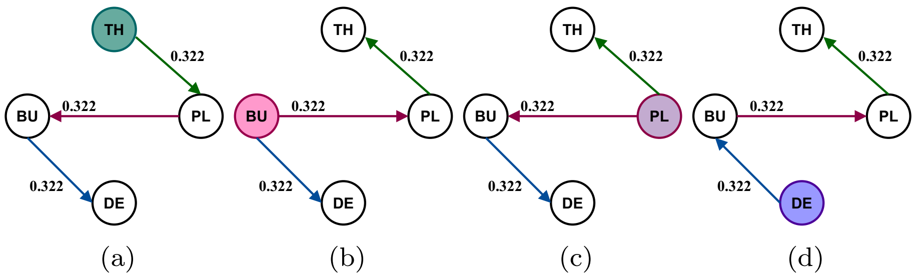


**Fig. 6** Assignment of Root nodes (a) Choose *Th* as root node and its LL= −5.033(b) Choose *BU* as root node and its LL= −5.033 (c) Choose *PL* as root node and its LL= −5.6925 (d) Choose *De* as root node and its LL= −7.359

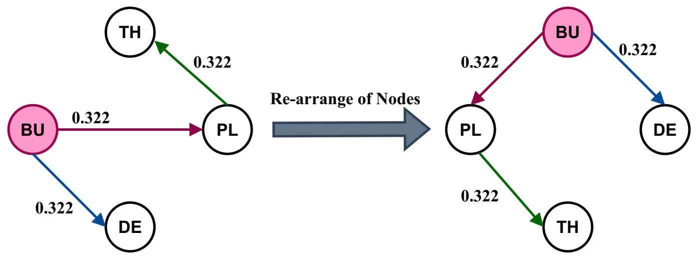


**Fig. 7** Considered *BU* as a root node, and adjusting the DAG for better representation

edges, as shown in Fig. 7. We consider Fig. 6b for further process of creating a learned representation for the Bayesian network.

Next, include the additional edges to the best DAG $D$ resulting from the chow-liu algorithm, without forming a cycle (to maintain DAG properties). Further, estimate the score of $D$ after adding each new edge using Eq. (2). e.g., the *Score* of the resultant DAG by chow-liu algorithm as shown in Fig. 8a is (−13.1598). We include the edges one by one and estimate the *Score*, as shown in Fig. 8. First, we add an edge between the nodes *DeE* to *TH* (*TH* to *DE* forms a cycle, so it is not valid) and estimate the score. Adding $(\overline{DE,TH})$ results in the score (−14.1357), and pictorial representation is shown in Fig. 8b. Next, we will add another edge to estimate the maximum score, and we get an added edge between *De* and *PL*, so resultant *Score* is (−14.1598), whose pictorial representation is shown in Fig. 8c. We further add another edge between *BU* and *Th*, and the estimated score here is (−15.2966), and the pictorial representation is depicted in Fig. 8d. Finally, we consider the best score achieved DAG, which is shown in Fig. 8a.

The learned representation using BNSL is shown in Fig. 9 with each node and its associated correlation values. In Fig. 9, the tabular entries for each node are considered based on the input dataset shown in Eq. (1). This information helps in predicting or monitoring the DCCS performance and helps to get the predictions, e.g., device events and probabilities for the four performance metrics in the DCCS can be measured using this data. E.g., a device added and its performance metrics may be considered as *TH* and *BU* are above the threshold, but the Delay and *PL* are under the threshold values. The probability correlation for this situation in the DCCS is measured according to the representation (see Fig. 9) as $Pr(TH, BU, \sim DE, \sim PL)$ which is the product of probabilities of $Pr(TH| \sim PL)$ ($PL$ is the parent to $TH$),

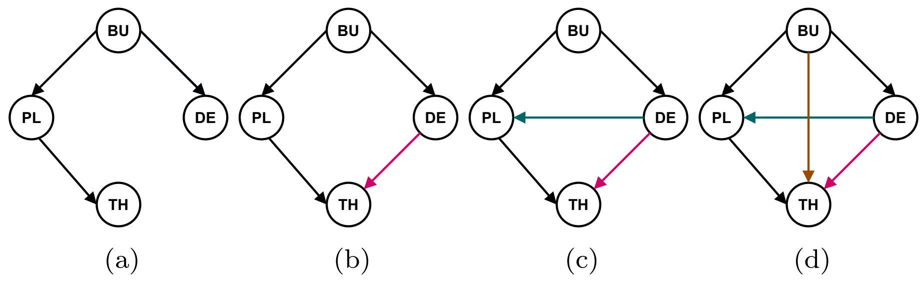
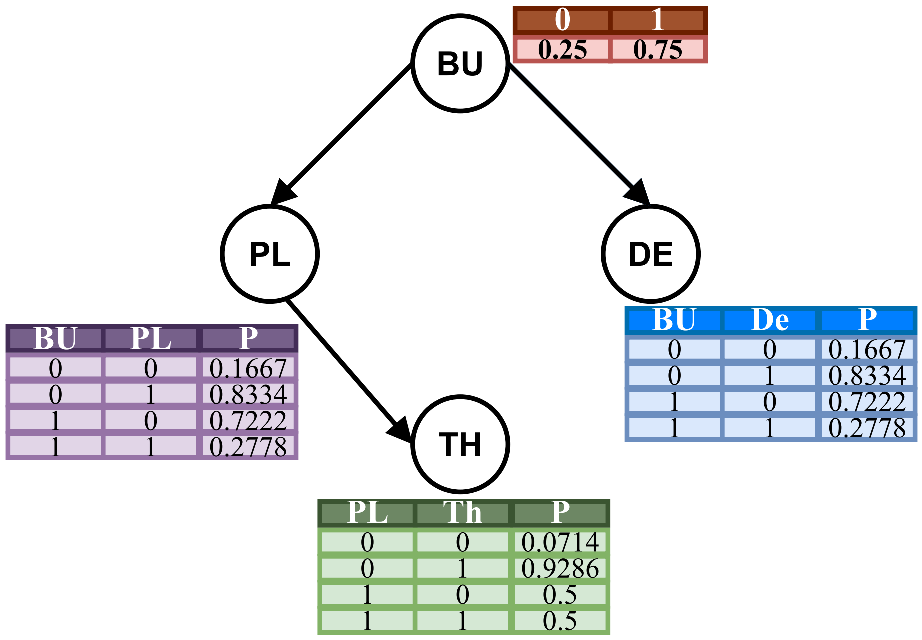


**Fig. 8** Maximum scored DAG for BNSL by adding the edges without forming cycles (a) Resultant DSG of chow-liu algorithm (Score = −13.1598) (b) Add an edge between *De* to *Th* (Score = −14.1357) (c) Add an edge between *De* to *PL* (Score = −14.1598) (d) Add an edge between *BU* to *Th* (Score = −15.2966)



| 0 | 1 |
|---|---|
| 0.25 | 0.75 |

| BU | PL | P |
|---|---|---|
| 0 | 0 | 0.1667 |
| 0 | 1 | 0.8334 |
| 1 | 0 | 0.7222 |
| 1 | 1 | 0.2778 |

| BU | De | P |
|---|---|---|
| 0 | 0 | 0.1667 |
| 0 | 1 | 0.8334 |
| 1 | 0 | 0.7222 |
| 1 | 1 | 0.2778 |

| PL | Th | P |
|---|---|---|
| 0 | 0 | 0.0714 |
| 0 | 1 | 0.9286 |
| 1 | 0 | 0.5 |
| 1 | 1 | 0.5 |

**Fig. 9** Learned Bayesian Network Structure learning for the data available at Eq. (1), and its correlation information which helps in predictions and monitoring

probabilities of $Pr(BU)$ ($BU$ is the root, so direct probability when it is 1), probabilities of $Pr(\sim DE| \sim BU)$ ($BU$ is the parent to $DE$), and probabilities of $Pr(\sim PL|BU)$ ($BU$ is the parent to $PL$). From Fig. 9, we notice the numerical values for and substitute them appropriately i.e., $0.9286 \times 0.75 \times 0.7222 \times 0.7222$. The probability that something occurs in the system is 0.36325.

Similarly, we can analyze the multiple correlations among these performance metrics using the learned representation. We have considered only a few metrics and data to gain insight into the algorithm, but if we considered multiple metrics and devices, we could have achieved even better correlations. Altering the rows and columns of the input data also helps in analyzing the performance of each device and we can achieve causal relation with other devices in the DCCS.

### Knowledge

The term “knowledge” in the DCCS governance and sustainability model refers to information that can be extracted from data through analysis or insight. An imperative determinant of the quality of the knowledge that has been acquired from learning data is its ability to be generalized. The accumulation of knowledge is the result of interactions with a pool of information. It is easier and faster to monitor, predict, and make decisions when knowledge is extracted. There are several ways to represent knowledge, but graph representation is an easy way to store and visualize [68]. The knowledge graph is a comprehensive view of the nodes, edges, and labels. On ingestion of data, they are capable of identifying individual objects and their relationships with one another [69]. An extended form of knowledge representation is the use of DAGs as illustrated in Fig. 9, which are easy to query and provide decisions or answers.

### Monitoring and prediction

DCCS involves a wide variety of devices (from the edge to the cloud), each with its capabilities. Hence, testing the performance of each device is essential to achieve maximum benefit. As there are so many devices present and their locations also vary, it is impossible to keep track of them manually. However, there are several monitoring tools in the literature (discussed in Sect. "Existing solutions or tools for IoT monitoring" ), but they are not useful for monitoring the entire DCCS. The governance and sustainability model is used to monitor the condition and efficiency of these devices autonomously, to maximize their benefit from them. Thus, the knowledge extracted from DCCS data helps identify the root cause of the problem, such as manufacturer’s defects, network issues, inefficient energy use, etc. The ready knowledge helps to identify or predict the issues in the devices and notify the administrator or try to self-heal to achieve the best usage of devices. As a result of readily available knowledge, we can identify or predict issues with devices and report them to the administration or attempt to self-heal to achieve maximum effectiveness. In addition to showing the issues, predictions and monitoring variables are also useful in improving the efficiency of resource utilization. Due to the limitations of the computational resources of edge devices, it is difficult to determine their availability for performing tasks [70, 71]. In addition to determining the most effective use of resources through prediction, the knowledge extracted from the data also assists in identifying the most appropriate future needs for the devices and their resources. However, it is necessary to verify the reliability of the suggestions made by governance and sustainable models, and this is a challenging issue. This challenge can be mitigated by leveraging SLOs [72, 73]. Whenever monitoring or predicting issues are identified through querying, their reliability must be verified by mapping them to SLOs before a decision is made.

### Notifications and/or visualization

In the governance process, notification plays a vital role in resolving errors in DCCS. However, our model monitors and predicts several aspects such as networking, computation efficiency, device health, etc., so it is crucial to decide how these notifications are delivered to administrators. These notifications are usually in text format and can be received via text messages, push notes, emails, or sent to visualization tools. In any case, it is necessary to set up who can receive these messages (privacy concerns), what frequency of notifications is appropriate, and what level of information to transmit. Multiple operations are running on DCCS simultaneously, and it is important to decide whether notifications should be delivered to relevant people who monitor specific aspects of DCCS. Privacy concerns make it necessary to restrict notifications to unauthorized entities.

Another significant metric to take into account is the frequency of notifications since more frequent notifications require more computational resources. However, continuous monitoring is a requirement of DCCS, so determining the frequency is crucial. It is imperative to know what level of detail should be notified to the administrator. Consequently, if the administrator needs to solve the issue, they must know the details of where the problem arises, or what the root cause of the issue is. Since the environment is dynamic, we need an adaptive framework, it will be decided dynamically depending on the issues raised. When there are no issues, report the performance metrics to the administrator without providing detailed analysis. When a device fails to respond, there is usually a detailed problem such as either a network failure, power supply failure or any other problem. Once the problem is identified, the administrator can solve it. To establish dynamic DCCS governance, it is necessary to make autonomous decisions based on these notifications. These notifications are further analyzed or compared using SLOs to decide if the notified problem can be rectified, or if it truly caused a problem or not.

The visualization can be a dashboard that provides analytical results in a graphical interface. Visualization provides a powerful means of monitoring DCCS performance in terms of various metrics, enabling the user to understand patterns and malfunctions more easily. These visualization metrics include network performance, computational efficiency, resource usage, etc. Visualization tools can alert administrators by highlighting the fault. Graphite and Grafana [74] are open-source tools that provide a visualization with flexible and custom dashboards. In the literature, these tools are used in MetricFire [41] for efficient visualizations. These tools are popular for visualization because of efficient storage in databases, having a web-based interface to interact with data and easy scripting for customization. Recent advances such as digital twins [75–77], AR/VR [78], and metaverse technologies [79–81] can also help improve DCCS visualization effectively. These tools are very efficient at identifying the fault and visualizing them to the user/administrator, but they are not able to solve the problem autonomously.

## Decision making

In the governance and sustainability model, the decision-making phase shows that some actions will be taken to correct problems identified in previous phases. These decisions are categorized into autonomous and manual decision-making. Autonomous decision-making in this context refers to the DCCS being able to rectify issues without human intervention. This can be achieved through predefined rules or using learning-driven approaches. When a rule-based approach is used, the notifications match the SLOs and make the necessary decisions. A learning-driven approach monitors DCCS behavior continuously while making dynamic decisions. Among these two approaches, SLOs provide quick decisions and balance time and computation, whereas learning-based methods are computationally intensive. However, combining both scenarios leads to better results and helps with dynamic decision-making. Manual decisions are made by the administrator based on identified problems that are not possible to address autonomously. E.g., network failure, hardware damage, being disconnected from the network through physical failures, etc. This means that problems that are not addressed through software instructions must be handled by humans or robots, but not autonomously by the system.

## Actions

Once the decision-making system recommends an autonomously solvable problem, it will be further received by the action stage to cure the failure or fault. Other issues are handled manually by the appropriate administrator. The autonomously solvable problems are solved similarly to the self-healing process of the human body (this paper considered wound healing). The human body evolved over millions of years to develop a complex and highly effective self-healing mechanism. The human body and DCCS have many similarities, and using this self-healing system not only helps improve performance but also minimizes managing costs and performance improvement. Simulating an similar system is extremely difficult but not impossible. However, it can be achieved through continuous monitoring and analysis of collected DCCS data.

The mechanism of self-healing in the Action phase of the governance and sustainable DCCS model is illustrated in Figure 10. The four stages of the wound healing process described in motivation (subsection ) are described concerning DCCS as follows.

| | |
|---|---|
| Hemostasis (Prevent from other damages): | When a fault or failure is reported by the decision-making system, this phase starts immediately. During this phase, services attached to a specific system are diverted to other nearby nodes. This includes rerouting, activating a backup process, or diverting computing tasks to other available systems. Therefore, future damages such as data loss and computation delays can be prevented. |

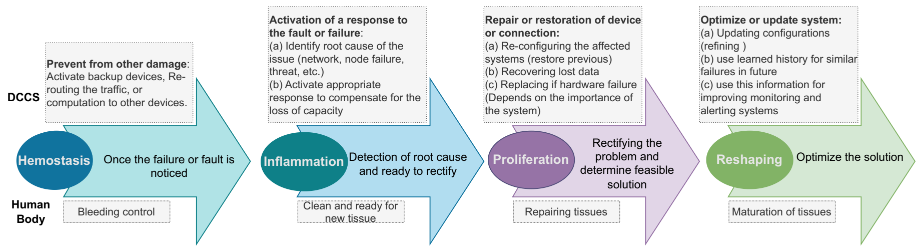

DCCS
Human Body
Prevent from other damage: Activate backup devices, Re-routing the traffic, or computation to other devices.
Hemostasis
Once the failure or fault is noticed
Bleeding control
Activation of a response to the fault or failure: (a) Identify root cause of the issue (network, node failure, threat, etc.) (b) Activate appropriate response to compensate for the loss of capacity
Inflammation
Detection of root cause and ready to rectify
Clean and ready for new tissue
Repair or restoration of device or connection: (a) Re-configuring the affected systems (restore previous) (b) Recovering lost data (c) Replacing if hardware failure (Depends on the importance of the system)
Proliferation
Rectifying the problem and determine feasible solution
Repairing tissues
Optimize or update system: (a) Updating configurations (refining ) (b) use learned history for similar failures in future (c) use this information for improving monitoring and alerting systems
Reshaping
Optimize the solution
Maturation of tissues

**Fig. 10** Action (Self-healing) taken place after the fail or fault is detected by decision-making system

| | |
|---|---|
| Inflammation (Detection of root cause): | During the inflammation phase, the system starts identifying the root cause of the problem, such as connection failure, system overload, buffer overflows, energy drain, untrusted requests to access resources, etc. Once the root cause is recognized, this phase determines the resources needed to solve the issue. |
| Proliferation (Repair or restoration of the problem): | During this phase, the model works on rectifying the issue and begins recovering lost data, including restoring configurations and working on resolving the issue. If the system cannot be recovered or restored, it recommends replacing it with the appropriate administrators. |
| Reshaping (Optimize or update system): | This phase involves sending the final decision back to the decision-making system to update the learning model so that further decisions can be optimized. Reconfigurations and system updates also fall under this phase. |

## Challenges

DCCS are designed to provide efficient computations for business data. But, it is necessary to govern (through monitoring tools) the devices to get longer sustainability with minimal downtime. These monitoring tools have to do additional analyses of the information from various sources. So, it is necessary to trade off the computations between the business and device data, which imposes several challenges. They are summarized as follows:

Priority: Since the primary goal of DCCS is to analyze business data, it is necessary to give it a high priority. However, as business data flows continuously in the DCCS, it is necessary to allocate computing slots or dedicated devices for governance. Since centralizing governance is not a promising strategy, it is necessary to consider the priorities of the workloads between business data computation and monitoring strategies.

| | |
|---|---|
| Starvation: | In addition to the above challenge, it is also necessary to avoid starvation while assigning priorities. If we give the highest priority to business data analysis, we must ensure that the governance process does not suffer from starvation, and vice versa. To avoid starvation, schedule monitoring tasks and business data processing by a deadline. |
| Consistency: | It is necessary to estimate, how frequently the governance process will take place. DCCS run dynamically, so it is necessary to monitor them consistently to mitigate downtime. However, this frequent system data analysis does not affect business data analysis. |
| Privacy: | Analytics are on log and configuration files during the monitoring process. Therefore, it is necessary to preserve privacy. |

## Future research directions

Governance and sustainability for DCCS is a conceptual framework, but it can be further developed using the following advances to become more dynamic and autonomous. The challenges and benefits of these advancements are also discussed here.

### Minimize computational load at edge

It is possible to perform monitoring tasks anywhere in DCCS, such as cloud, fog, or edge. But, performing this task at the Edge is more beneficial in terms of shortening the delay in providing services. As AI and ML strategies have become more widespread and beneficial, most edge devices are using them to some extent. Edge intelligence has become popular because of its minimal latency and quick decision-making. Since AI/ML approaches are computationally hungry, they consume more computation and use more energy and memory [82]. Due to the rapid power drain of edge nodes, they can only sustain for a short time. Hence, it is pertinent to determine whether AI/ML is needed for edge devices and minimize unnecessary computations as much as possible. It does not mean stopping the use of AI/ML techniques at the edge, but choosing the most appropriate algorithms based on the application, which can be efficient in the trade-off between efficiency and long-term sustainability. There are several ways to minimize edge computations, mainly when learning algorithms are run. This includes 1) distribution of learning among multiple edge devices [83, 84], 2) skipping some parts of the interactions that are unnecessary, e.g. frame skipping methods in deep neural networks [85], 3) provide input as knowledge or features [86], 4) decide on an optimal number of layers sufficient to produce maximum accuracy [87], etc. The majority of learning models extract correlations between variables but do not extract causal relationships. So when comparing causality with correlation, causality results in better predictions. In addition, causality approaches are computationally more efficient than correlation approaches. So, there is a chance to minimize computations at the edge through causality.

### Learning multiple tasks

DCCS are an example of a system in which multiple related but not identical tasks are performed simultaneously. E.g., the information on a device configuration and its network configuration is not identical but related. Learning such tasks can simultaneously improve the system's overall performance and also accuracy [88]. Multi-task Learning (MTL) strategies can optimize multiple objective functions simultaneously through learning different tasks. These approaches initially classify the jobs from the given data and perform learning separately. In the computing continuum, MTL is used to identify inference attacks [89], traffic predictions in the IoT [90], computational offloading at Edge [91], among others. Especially through MTL, different tasks, such as device monitoring, network monitoring, and privacy preservation, can simultaneously perform operations from the available system data. Since we are working with resource-constrained devices in DCCS, distributed MTL can be more promising [92]. It can minimize the load on a single machine, and the tasks/learning models are distributed among multiple devices.

### Zero-touch provisioning

Zero-touch Provisioning (ZTP) is an end-to-end network service provisioning model to configure network devices [93, 94]. The devices in DCCS software and configurations need to be updated in a timely manner. There are several heterogeneous devices that make it difficult for humans to perform it. In order to fill this gap, software services like ZTP can be used in governance and sustainable frameworks. The ZTP is used for configuration management in the literature for several network-based services. However, the current functionalities of the ZTP could be improved [95]. Extending these services can help the user or administrators rectify issues diagnosed through the proposed reference model. Currently, the ZTP services are running through centralized cloud servers, but it is necessary to decentralize them. When working with a large to a vast number of computing devices, the distributed ZTP increases its services and rapidly completes its assigned tasks. Consequently, by using ZTP, DCCS becomes more autonomous and is able to reduce failures or response delays without the involvement of humans.

### Privacy preserving

Efficient DCCS monitoring requires the use of diversified information from each device that includes sensitive configuration details. The configuration files are sensitive because they contain information such as passwords, access keys, or several authors' information that cannot be shared with unauthorized parties. When the models we propose are considered to learn or monitor devices, each piece of information can determine the quality of the monitoring. So, an efficient monitoring tool can also protect our data and must follow privacy rules. Misusing this information can cause a device or network to malfunction. An efficient way to access the configuration details of any device is to use encryption mechanisms [96]. There are several secure data or file transmission approaches available in the literature, such as Secure FTP (SFTP/FTPS), and Secure Shell (SSH), which enable authentication mechanisms before accessing the information for learning, version control, etc. The learning process performs only read operations to learn from the given input data to monitor the systems, but there is a chance that secret information can be misused when privacy is not enabled. Zero-trust architecture is a solution that can protect accessing such sensitive information [97, 98].

### Reflect the human-like healing system

From Subsection , we understand how the human body as a complex system can heal itself to sustain the human body longer. Since the behavior and complex functionality of DCCS resemble the human body [21], there is a chance to reflect similar healing features in DCCS. However, understanding the human body is a challenging task compared to the complex behavior of DCCS. Therefore, the information from the DCCS can help identify the root cause of the problem in the systems and further use this information (see Sect. "Source of DCCS's data") to heal a problem without external instructions or human interventions. Using efficient, lightweight, and accurate learning algorithms that can understand the underlying information from DCCS's data will further help in understanding the root cause. Understanding the root cause of any problem helps to fix the problems easily. However, deciding the accuracy of the identified cause is a challenging issue.

## Conclusion

Generally, DCCS techniques are geared towards computing business data efficiently, instead of focusing on sustainability through adequate monitoring and system condition prediction. But only a healthy DCCS will be able to provide efficient services and endure. This paper focuses on the data associated with the computing devices in DCCS, and how it helps to govern and achieve sustainability with zero downtime. This paper first describes the various formats of data that are available for each device in DCCS. Further, we described the complexities inherent in these data due to the heterogeneity of the computing devices. Furthermore, it is necessary to associate these data with the characteristics of big data, so that their analytics tools can be used to analyze it. In the present literature, there exist only tools for monitoring IoT devices that are analyzed and provided with their pitfalls, hence, it lacks a tool or model for monitoring or predicting the condition of an entire DCCS. To solve this, we introduced a governance and sustainability model, which uses the device data. For monitoring and predicting DCCS conditions, data can be transformed through a variety of stages, such as learning, knowledge, notifications, and decision-making. We examine an illustration to gain a better understanding of how the model works. Finally, this article summarizes the limitations and possible extensions of an efficient DCCS governance model to maintain zero downtime over the long term.

**Abbreviations**

| | |
|---|---|
| AI | Artificial intelligence |
| API | Application program interface |
| AWS | Amazon web services |
| BIC | Bayesian information criterion |
| BNSL | Bayesian network structure learning |
| CRL | Contrastive representation learning |
| CSV | Comma-separated values |
| DAG | Directed acyclic graphs |
| DCCS | Distributed compute continuum systems |
| GPS | Global positioning system |
| GPU | Graphics processing unit |

| | |
|---|---|
| GRL | Graph representation learning |
| GUI | Graphical user interface |
| HDMI | High definition multimedia interface |
| HTTP | Hypertext transfer protocol |
| IoT | Internet of things |
| JSON | JavaScript object notation |
| MI | Mutual information |
| MQTT | Message queue telemetry transport |
| ML | Machine learning |
| MTL | Multi-task learning |
| NoSQL | Not only structure query language |
| NLP | Natural language processing |
| ReL | Representation learning |
| RF | Ratio frequency |
| RWL | Random walk learning |
| SFTP | Secure file transfer protocol |
| SLO | Service-level objective |
| SSH | Secure shell |
| XML | eXtensible Markup Language |
| ZTP | Zero-touch provisioning |

### Author contributions
All the listed authors in this article are contributed equally. All authors read and approved the final manuscript.

### Funding
Research has received funding from the EU's Horizon Europe Research and Innovation Program under Grant Agreement No. 101070186. EU website for Teadal: https://www.teadal.eu/

### Availability of data and materials
All the data related to this article is either discussed or cited with in the paper.

## Declarations

### Ethics approval and consent to participate
Not Available.

### Competing interests
The authors declare that they have no competing interests.

Received: 31 January 2023 Accepted: 14 April 2023
Published online: 28 April 2023

### References
1. Balouek-Thomert D, Renart EG, Zamani AR, Simonet A, Parashar M. Towards a computing continuum: enabling edge-to-cloud integration for data-driven workflows. Int J High Perform Comput Appl. 2019;33(6):1159–74. https://doi.org/10.1177/1094342019877383.
2. Dustdar S, Pujol VC, Donta PK. On distributed computing continuum systems. IEEE Trans Knowl Data Eng. 2023;35(4):4092–105. https://doi.org/10.1109/TKDE.2022.3142856.
3. Morichetta A, Pujol VC, Dustdar S. A roadmap on learning and reasoning for distributed computing continuum ecosystems. In: 2021 IEEE International Conference on Edge Computing (EDGE), 2021;pp. 25–31.
4. Beckman P, Dongarra J, Ferrier N, Fox G, Moore T, Reed D, Beck M. Harnessing the computing continuum for programming our world. In: Zomaya A, Abbas A, Khan S, editors. Fog computing: theory and practice. Hoboken: Wiley; 2020. p. 215–30.
5. Balouek-Thomert D, Rodero I, Parashar M. Harnessing the computing continuum for urgent science. ACM SIGMETRICS Perform Eval Rev. 2020;48(2):41–6.
6. Avasalcai C, Murturi I, Dustdar S. Edge and fog: a survey, use cases, and future challenges. In: Zomaya A, Abbas A, Khan S, editors. Fog computing: theory and practice. Hoboken: Wiley; 2020. p. 43–65.
7. Liu G, Dai F, Xu X, Fu X, Dou W, Kumar N, Bilal M. An adaptive DNN inference acceleration framework with end-edge-cloud collaborative computing. Future Gener Comput Syst. 2023;140:422–35.
8. Singh R, Kovacs J, Kiss T. To offload or not? an analysis of big data offloading strategies from edge to cloud. In: 2022 IEEE World AI IoT Congress (AIIoT), 2022;p. 046–052.
9. Hong Z, Chen W, Huang H, Guo S, Zheng Z. Multi-hop cooperative computation offloading for industrial IoT-edge-cloud computing environments. IEEE Trans Parallel and Distrib Syst. 2019;30(12):2759–74.
10. Robles-Enciso A, Skarmeta AF. A multi-layer guided reinforcement learning-based tasks offloading in edge computing. Comput Netw. 2023;220: 109476.
11. Zobaed S. AI-driven confidential computing across edge-to-cloud continuum. arXiv. 2023. https://doi.org/10.48550/arXiv.2301.00928.
12. Casamayor-Pujol V, Morichetta A, Ilir M, Donta PK, Dustdar S. Fundamental research challenges for distributed computing continuum systems. Information. 2023;2023:1–7.

13. Firouzi F, Farahani B, Marinšek A. The convergence and interplay of edge, fog, and cloud in the AI-driven internet of things (IoT). Inform Syst. 2022;107: 101840.
14. Pusztai T, Morichetta A, Pujol VC, Dustdar S, Nastic S, Ding X, Vij D, Xiong Y. Slo script: A novel language for implementing complex cloud-native elasticity-driven slos. In: 2021 IEEE International Conference on Web Services (ICWS), 2021;p. 21–31.
15. Nastic S, Morichetta A, Pusztai T, Dustdar S, Ding X, Vij D, Xiong Y. SLOC: service level objectives for next generation cloud computing. IEEE Internet Comput. 2020;24(3):39–50.
16. Fu-Kiau KKB. Self-healing power and therapy. Baltimore: Black Classic Press; 1991.
17. Pollack SV. The wound healing process. Clin Dermatol. 1984;2(3):8–16.
18. Guo SA, Dipietro LA. Factors affecting wound healing. J Dent Res. 2010;89(3):219–29.
19. Kirsner RS, Eaglstein WH. The wound healing process. Dermatol Clin. 1993;11(4):629–40.
20. Enoch S, Leaper DJ. Basic science of wound healing. Surgery. 2005;23(2):37–42.
21. Casamayor Pujol V, Donta PK, Morichetta A, Murturi I, Dustdar S. Distributed computing continuum systems–opportunities and research challenges. In: Service-Oriented Computing–ICSOC 2022 Workshops: ASOCA, AI-PA, FMCIoT, WESOACS 2022, Sevilla, Spain, November 29–December 2, 2022 Proceedings, 2023;pp. 405–407. Springer.
22. Roman D, Prodan R, Nikolov N, Soylu A, Matskin M, Marrella A, Kimovski D, Elvesæter B, Simonet-Boulogne A, Ledakis G. Big data pipelines on the computing continuum: tapping the dark data. Computer. 2022;55(11):74–84.
23. Donta PK, Dustdar S. The promising role of representation learning for distributed computing continuum systems. In: 2022 IEEE International Conference on Service-Oriented System Engineering (SOSE), 2022;p. 126–132.
24. Landauer M, Skopik F, Frank M, Hotwagner W, Wurzenberger M, Rauber A. Maintainable log datasets for evaluation of intrusion detection systems. arXiv. 2022. https://doi.org/10.48550/arXiv.2203.08580.
25. Alsaedi A, Moustafa N, Tari Z, Mahmood A, Anwar A. Ton_IoT telemetry dataset: a new generation dataset of IoT and IIoT for data-driven intrusion detection systems. IEEE Access. 2020;8:165130–50. https://doi.org/10.1109/ACCESS.2020.3022862.
26. Moldovan D, Copil G, Truong H-L, Dustdar S. MELA: Monitoring and analyzing elasticity of cloud services. In: 2013 IEEE 5th International Conference on Cloud Computing Technology and Science, 2013;vol. 1, p. 80–87.
27. Chi Y, Da Costa MF. Harnessing sparsity over the continuum: atomic norm minimization for superresolution. IEEE Signal Process Mag. 2020;37(2):39–57. https://doi.org/10.1109/MSP.2019.2962209.
28. Montori F, Jayaraman PP, Yavari A, Hassani A, Georgakopoulos D. The curse of sensing: survey of techniques and challenges to cope with sparse and dense data in mobile crowd sensing for internet of things. Pervasive Mob Comput. 2018;49:111–25. https://doi.org/10.1016/j.pmcj.2018.06.009.
29. Bughin J. Big data, big bang? J Big Data. 2016;3(1):1–14.
30. Talia D. Clouds for scalable big data analytics. Computer. 2013;46(05):98–101.
31. Buhl HU, Röglinger M, Moser F, Heidemann J. Big data. Berlin: Springer; 2013. https://doi.org/10.1007/s12599-013-0249-5.
32. Wolfert S, Ge L, Verdouw C, Bogaardt M-J. Big data in smart farming-a review. Agric Syst. 2017;153:69–80.
33. Géczy P. Big data characteristics. Macrotheme Rev. 2014;3(6):94–104.
34. Sun Z, Strang K, Li R. Big data with ten big characteristics. In: Proceedings of the 2nd International Conference on Big Data Research, 2018;p. 56–61.
35. Kolajo T, Daramola O, Adebiyi A. Big data stream analysis: a systematic literature review. J Big Data. 2019;6(1):1–30.
36. Roman D, Nikolov N, Soylu A, Elvesæter B, Song H, Prodan R, Kimovski D, Marrella A, Leotta F, Matskin M. Big data pipelines on the computing continuum: Ecosystem and use cases overview. In: 2021 IEEE Symposium on Computers and Communications (ISCC), 2021;p. 1–4. .
37. O'donovan P, Leahy P, Bruton K, O'Sullivan DT. Big data in manufacturing: a systematic mapping study. J Big Data. 2015;2(1):1–22.
38. Domotz: Unparalleled network monitoring software. https://www.domotz.com/. Accessed 17 Apr 2023.
39. Datadog: Monitor IoT performance and availability for distributed device fleets. https://www.datadoghq.com/solutions/iot-monitoring/. Accessed 17 Apr 2023.
40. Particle: particle: reprogram the World. https://www.particle.io/. Accessed 17 Apr 2023.
41. MetricFire: MetricFire: hosted infrastructure and application monitoring. https://www.metricfire.com/. Accessed 17 Apr 2023.
42. ThingWorx: build a better IIoT solution with ThingWorx. https://www.ptc.com/en/products/thingworx. Accessed 17 Apr 2023.
43. Splunk: splunk: the basics of IoT monitoring. https://www.splunk.com/en_us/iot/monitoring-and-diagnostics.html. Accessed 17 Apr 2023.
44. Senseye: Senseye PdM. https://www.senseye.io/. Accessed 17 Apr 2023.
45. SkySpark: SkySpark. https://skyfoundry.com/. Accessed 17 Apr 2023.
46. Oracle: Oracle IoT. https://www.oracle.com/internet-of-things/. Accessed 17 Apr 2023.
47. Amazon: AWS IoT device monitoring. https://aws.amazon.com/iot-device-management/. Accessed 17 Apr 2023.
48. Salesforce: Salesforce IoT cloud. https://www.salesforce.com/products/field-service/overview/. Accessed 17 Apr 2023.
49. Microsoft: Microsoft Azure IoT Suite. https://azure.microsoft.com/en-gb/products/iot-hub/. Accessed 17 Apr 2023.
50. Watson I. IBM Watson IoT. https://www.ibm.com/cloud/internet-of-things. Accessed 17 Apr 2023.
51. TeamViewer: TeamViewer IoT. https://www.teamviewer.com/en-us/teamviewer-tensor/enable-iot-platforms/. Accessed 17 Apr 2023.
52. Donta PK, Srirama SN, Amgoth T, Annavarapu CSR. Survey on recent advances in IoT application layer protocols and machine learning scope for research directions. Digit Commun Netw. 2022;8(5):727–44. https://doi.org/10.1016/j.dcan.2021.10.004.
53. Rusu O, Halcu I, Grigoriu O, Neculoiu G, Sandulescu V, Marinescu M, Marinescu V. Converting unstructured and semi-structured data into knowledge. In: 2013 11th RoEduNet International Conference, 2013;p. 1–4.

54. Rosett CM, Hagerty A. Data wrangling. In: Rosett CM, Hagerty A, editors. Introducing HR analytics with machine learning. Berlin: Springer; 2021. p. 217–41.
55. Furche T, Gottlob G, Libkin L, Orsi G, Paton N. Data wrangling for big data: Challenges and opportunities. In: Advances in Database Technology-EDBT 2016: Proceedings of the 19th International Conference on Extending Database Technology, 2016;p. 473–478.
56. Wu P, Lu Z, Zhou Q, Lei Z, Li X, Qiu M, Hung PC. Bigdata logs analysis based on seq2seq networks for cognitive internet of things. Future Gener Comput Syst. 2019;90:477–88.
57. Subramaniyaswamy V, Vijayakumar V, Logesh R, Indragandhi V. Unstructured data analysis on big data using map reduce. Procedia Comput Sci. 2015;50:456–65.
58. García S, Luengo J, Herrera F. Data preprocessing in data mining. Berlin: Springer; 2015. p. 72.
59. Schölkopf B, Locatello F, Bauer S, Ke NR, Kalchbrenner N, Goyal A, Bengio Y. Toward causal representation learning. Proc IEEE. 2021;109(5):612–34.
60. Zhang D, Yin J, Zhu X, Zhang C. Network representation learning: a survey. IEEE Trans Big Data. 2018;6(1):3–28.
61. Xie Y, Yu B, Lv S, Zhang C, Wang G, Gong M. A survey on heterogeneous network representation learning. Pattern Recognit. 2021;116: 107936.
62. Bengio Y, Courville A, Vincent P. Representation learning: a review and new perspectives. IEEE Trans Pattern Anal Mach Intell. 2013;35(8):1798–828. https://doi.org/10.1109/TPAMI.2013.50.
63. Tsamardinos I, Brown LE, Aliferis CF. The max-min hill-climbing Bayesian network structure learning algorithm. Mach Learn. 2006;65(1):31–78.
64. Scanagatta M, Salmerón A, Stella F. A survey on Bayesian network structure learning from data. Prog Artif Intell. 2019;8(4):425–39.
65. Rusek J, Tajduś K, Firek K, Jedrzejczyk A. Score-based Bayesian belief network structure learning in damage risk modelling of mining areas building development. J Clean Prod. 2021;296: 126528.
66. Suzuki J. The bayesian chow-liu algorithm. In: The Sixth European Workshop on Probabilistic Graphical Models, 2012;p. 315–322.
67. De Leeuw J. Correctness of kruskal's algorithms for monotone regression with ties. Psychometrika. 1977;42(1):141–4.
68. Portmann E, Kaltenrieder P, Pedrycz W. Knowledge representation through graphs. Procedia Comput Sci. 2015;62:245–8. https://doi.org/10.1016/j.procs.2015.08.446.
69. Hogan A, Blomqvist E, Cochez M, d'Amato C, Melo Gd, Gutierrez C, Kirrane S, Gayo JEL, Navigli R, Neumaier S. Knowledge graphs. ACM Comput Surv (CSUR). 2021;54(4):1–37.
70. Hazra A, Donta PK, Amgoth T, Dustdar S. Cooperative transmission scheduling and computation offloading with collaboration of fog and cloud for industrial IoT applications. IEEE Internet Things J. 2022. https://doi.org/10.1109/JIOT.2022.3150070.
71. Lan D, Taherkordi A, Eliassen F, Liu L, Delbruel S, Dustdar S, Yang Y. Task partitioning and orchestration on heterogeneous edge platforms: the case of vision applications. IEEE Internet Things J. 2022;9(10):7418–32.
72. Pusztai T, Morichetta A, Pujol VC, Dustdar S, Nastic S, Ding X, Vij D, Xiong Y. A novel middleware for efficiently implementing complex cloud-native SLOs. In: 2021 IEEE 14th International Conference on Cloud Computing (CLOUD), 2021;p. 410–420.
73. Nastic S, Pusztai T, Morichetta A, Pujol VC, Dustdar, S, Vii D, Xiong Y. Polaris scheduler: Edge sensitive and slo aware workload scheduling in cloud-edge-iot clusters. In: 2021 IEEE 14th International Conference on Cloud Computing (CLOUD), 2021;p. 206–216.
74. Chakraborty M, Kundan AP. Grafana. In: Chakraborty M, Kundan AP, editors. Monitoring cloud-native applications. Berlin: Springer; 2021. p. 187–240.
75. Tao F, Xiao B, Qi Q, Cheng J, Ji P. Digital twin modeling. J Manuf Syst. 2022;64:372–89.
76. Tao F, Qi Q. Make more digital twins. London: Nature Publishing Group; 2019. https://doi.org/10.1038/d41586-019-02849-1.
77. El Saddik A. Digital twins: the convergence of multimedia technologies. IEEE Multimed. 2018;25(2):87–92.
78. Gandhi HA, Jakymiw S, Barrett R, Mahaseth H, White AD. Real-time interactive simulation and visualization of organic molecules. Washington: ACS Publications; 2020. https://doi.org/10.1021/acs.jchemed.9b01161.
79. Cai Y, Llorca J, Tulino AM, Molisch AF. Compute-and data-intensive networks: the key to the metaverse. arXiv. 2022. https://doi.org/10.48550/arXiv.2204.02001.
80. Zawish M, Dharejo FA, Khowaja SA, Dev K, Davy S, Qureshi NMF, Bellavista P. AI and 6g into the metaverse: fundamentals, challenges and future research trends. arXiv. 2022. https://doi.org/10.48550/arXiv.2208.10921.
81. Hassanzadeh M. Metaverse and the fate of information systems. Sci Tech Inform Manag. 2022;8(1):7–14.
82. Murturi I, Egyed A, Dustdar S. Utilizing AI planning on the edge. IEEE Internet Comput. 2022;26(2):28–35.
83. Guo Y, Zhao R, Lai S, Fan L, Lei X, Karagiannidis GK. Distributed machine learning for multiuser mobile edge computing systems. IEEE J Sel Top Signal Process. 2022. https://doi.org/10.1109/JSTSP.2022.3140660.
84. Filho CP, Marques E Jr, Chang V, Dos Santos L, Bernardini F, Pires PF, Ochi L, Delicato FC. A systematic literature review on distributed machine learning in edge computing. Sensors. 2022;22(7):2665.
85. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G. Human-level control through deep reinforcement learning. Nature. 2015;518(7540):529–33.
86. Domingos P. A few useful things to know about machine learning. Commun ACM. 2012;55(10):78–87.
87. Lippmann R. An introduction to computing with neural nets. IEEE ASSP Mag. 1987;4(2):4–22.
88. Zhang Y, Yang Q. An overview of multi-task learning. Natl Sci Rev. 2018;5(1):30–43.
89. Ma X, Ma J, Kumari S, Wei F, Shojafar M, Alazab M. Privacy-preserving distributed multi-task learning against inference attack in cloud computing. ACM Trans Internet Technol (TOIT). 2021;22(2):1–24.
90. Wang S, Nie L, Li G, Wu Y, Ning Z. A multi-task learning-based network traffic prediction approach for sdn-enabled industrial internet of things. IEEE Transactions on Industrial Informatics 2022.
91. Yang B, Cao X, Bassey J, Li X, Qian L. Computation offloading in multi-access edge computing: a multi-task learning approach. IEEE trans Mob Comput. 2020;20(9):2745–62.

92. Wang J, Kolar M, Srerbo N. Distributed multi-task learning. In: Gretton, A, Robert, CC. (eds) Proceedings of the 19th International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, 2016;51:751–760. PMLR, Cadiz, Spain
93. Benzaid C, Taleb T. Ai-driven zero touch network and service management in 5g and beyond: challenges and research directions. IEEE Netw. 2020;34(2):186–94.
94. Angui B, Corbel R, Rodriguez VQ, Stephan E. Towards 6G zero touch networks: The case of automated Cloud-RAN deployments. In: 2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC), 2022;1–6. https://doi.org/10.1109/CCNC49033.2022.9700507.
95. Gallego-Madrid J, Sanchez-Iborra R, Ruiz PM, Skarmeta AF. Machine learning-based zero-touch network and service management: a survey. Digit Commun Netw. 2021. https://doi.org/10.1016/j.dcan.2021.09.001.
96. Subahi A, Theodorakopoulos G. Detecting IoT user behavior and sensitive information in encrypted IoT-app traffic. Sensors. 2019;19(21):4777.
97. Ahmed I, Nahar T, Urmi SS, Taher KA. Protection of sensitive data in zero trust model. In: Proceedings of the International Conference on Computing Advancements, 2020;1–5.
98. Ferretti L, Magnanini F, Andreolini M, Colajanni M. Survivable zero trust for cloud computing environments. Comput Secur. 2021;110: 102419.

## Publisher's Note