applied sciences

MDPI

Review

Federated Load Balancing in Smart Cities: A 6G, Cloud, and Agentic AI Perspective

Rohin Gillgallon, Giacomo Bergami * and Graham Morgan

School of Computing, Faculty of Science, Agriculture and Engineering, Newcastle University, Newcastle upon Tyne NE4 5TG, UK; r.gillgallon@newcastle.ac.uk (R.G.)
* Correspondence: bergamigiacomo@gmail.com or giacomo.bergami@newcastle.ac.uk

Abstract

Modern smart cities are comprised of multiple sensors, all with their own collection of communicating devices transmitting data towards cloud data centres for analysis. Smart cities have limited bandwidth resources, which, if not managed correctly, can lead to network bottlenecks. These bottlenecks are commonly addressed through bottleneck mitigation strategies and load balancing algorithms, which aim to maximise the throughput of a smart city's network infrastructure. Network simulators are a crucial tool for developing and testing bottleneck mitigation and load balancing techniques before deployment in real systems; however, many network simulators are developed as single-purpose tools, aiming to simulate a particular subset of an overarching use case. Such tools are therefore unable to model a real-world smart city infrastructure, which receives communications across a wide range of scenarios and from a wide variety of devices. This paper surveys the current state-of-the-art for network simulation tools, modern bottleneck mitigation strategies and load balancing techniques, evaluating each in terms of its suitability for smart cities and smart city simulation. This survey finds there is a lack of current network simulation tools up to the task of modelling smart city infrastructure and found no such simulation tools capable of modelling both smart city infrastructure and implementing the state-of-the-art bottleneck mitigation and load balancing strategies outlined within this work, highlighting this as a significant gap in current research before providing future work suggestions, including a federated approach for future simulation tools.

Keywords: Agentic AI; osmotic simulator; cloud simulator; edge simulator; IoT simulator; VANET Simulator; SimulatorOrchestrator

check for updates

Academic Editors: David Sarabia-Jácome and Carlos Enrique Palau Salvador

Received: 25 September 2025
Revised: 7 October 2025
Accepted: 9 October 2025
Published: 11 October 2025

Citation: Gillgallon, R.; Bergami, G.; Morgan, G. Federated Load Balancing in Smart Cities: A 6G, Cloud, and Agentic AI Perspective. *Appl. Sci.* **2025**, *15*, 10920. https://doi.org/10.3390/app152010920

1. Introduction

Smart cities, through the use of Internet of Things (IoT) technologies, aim to help achieve their sustainability aims [1], collecting data from across various city see slrs. The IoT technologies are used to both collect this data and then forward the data through a city's network infrastructure to data centres, where the data can be analysed appropriately [2]. Developing and managing the network infrastructure on the scale of a smart city is a difficult task, as not only are there multiple interconnected technologies working together, but the resulting network needs to be both robust and reliable through a wide variety of scenarios to ensure the secure and efficient exchange in information between various components within a city's network infrastructure to provide a given service adequately [3]. Given this complexity, network simulators are a vital tool in the development of network architectures along with network protocols [4]. However, most modern network simulators are focused on a particular aspect of the network infrastructure (such as the cloud or the edge) and do not allow for the complete modelling of a whole smart city's network infrastructure.

The key advantage of modelling a complete smart city in a single simulator is that load-balancing solutions (and bottleneck-mitigation strategies) can be tested with the whole network load of a smart city. As outlined in [5], it is not suitable to run one simulator after another (e.g., for pedestrians, general vehicles, ambulances, or smart devices) and then combine their respective outputs. This is also an unrealistic assumption, as each simulation will impact the others and, therefore, also the real-time behaviour of any load-balancing solutions in place. These complex interactions must be orchestrated to ensure that smart cities are equipped to handle the variable network load that can be expected in a city with a large population and numerous communicating devices (Section 5.3). Smart cities can often expect large influxes of people to events such as concerts and sporting events; these events will increase the number of communicating devices in the city and, therefore, also the strain on the city's network infrastructure (Section 1.1). To prevent cities' infrastructures from becoming overloaded in such circumstances requires a federated load-balancing approach, with a collection of load-balancing approaches and bottleneck-mitigation techniques all working together to keep up with a whole city's expected network load.

1.1. Use Case Scenario

In modern smart cities, patients can be provided with smart healthcare monitoring devices [6] which can transmit their health data to a data centre stored within a hospital, with the healthcare information accessible by their doctors [7]. If a patient's health begins to deteriorate, an ambulance can be dispatched to transport them to a hospital [8,9]. The benefits of the smart healthcare devices are removing the wait until a patient collapses, triggering a medical alert bracelet, or failing that, until they are found by someone who can ring for an ambulance on their behalf. This requires streaming the patients' data promptly; any delay would negate the time-saving benefit. However, within a smart city, there are devices unrelated to the service above that also transmit data through the city's infrastructure [5]. For example, vehicles may be transmitting their data as part of an intelligent transportation system as part of the city's scheme to reduce congestion and/or air pollution. A smart city's infrastructure needs to maintain a high Quality of Service (QoS) for the healthcare patients whilst managing the data traffic from the vehicles. Additionally, the smart city can use the data from the vehicle traffic to help the ambulance reach the patient as quickly as possible, optimising its route through busy city traffic, saving even more time, and improving the patient's healthcare outcomes. This can be achieved in two ways: either through AI-controlled vehicles or by simply providing a route for a human driver. This scenario, outlined within Figure 1, is considered to be sufficiently broad as to almost generalise to a full smart city simulation. The network traffic is routed from the IoT devices through an Software Defined Network (SDN)/Software-Defined Wide Area Network (SD-WAN) architecture to cloud hosts in a cloud data centre. The use of SD-WANs within smart cities allows for the control of the city infrastructure across multiple data centres [10]. Smart cities require the management of their vast number of IoT devices (e.g., pedestrians, smart devices, smart vehicles, and smart ambulances for healthcare scenarios). SD-WAN-based architectures allow for this management through the use of software via SD-WAN controllers.

The missing aspect not explicitly within Figure 1 is city resource utilisation. The most important resources for smart city infrastructures (and their simulation) are infrastructure energy usage, IoT device energy usage, vehicle energy usage, and network resource usage (e.g., bandwidth usage). The simulators discussed within the following sections will be assessed with respect to both how much of the scenario outlined in Figure 1 each simulator can model, as well as their ability to model city resource utilisation, and how well the simulator can model those aspects which are supported.

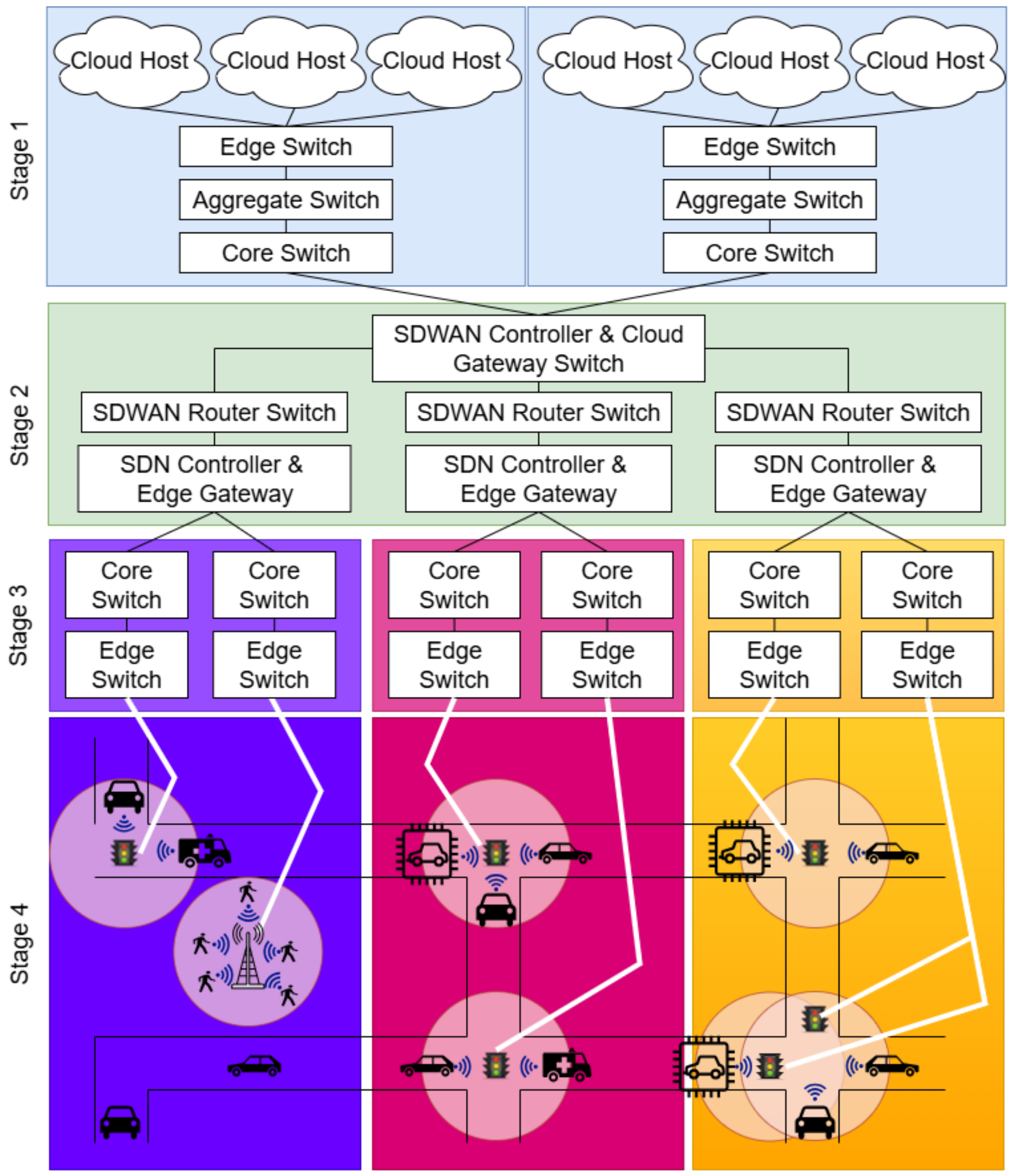

Figure 1. Example smart city SD-WAN architecture with people and cars, both regular and AI-controlled (shown by the car in the computer chip), along with ambulances all transmitting communications towards cloud hosts in cloud data centres (represented by the blue rectangles in Stage 1). The traffic lights and antennae are first-mile edge nodes that receive communications from nearby IoT devices within their communication radii (shown by the circles). The purple, magenta and yellow regions (Stages 3–4) represent different edge networks; devices communicating within these regions forward communications towards the nearest edge switch via the nearest first-mile edge node. The green region (Stage 2) represents an SD-WAN-controlled backbone network connecting all the edge networks to the cloud data centres.

Simulating such a scenario provides substantial differences from other simulators within other domains, e.g., electrical engineering. Works such as [11,12] both simulate the flow of energy from multiple sources in large-scale systems. However, such simulation tools cannot be repurposed to model the flow of data from multiple IoT devices to cloud data centres. Whilst in both scenarios there is the flow of a specific resource (data or energy) which can be ‘over-assigned’ to a given transmission link leading to congestion, as with [11], the flow of the resources is vastly different. Within an SDN SD-WAN infrastructure, data transmission functions through the preallocation of data for the various hops in the overall transmission path from IoT device to cloud data centre (see Figure 1). This is done to ensure the correct packets are dispatched to the correct locations. Energy (electricity), on the other hand, simply flows through wires, with no assignment of a ‘particular energy’ needing to reach a particular destination, as is the case with data packets. There is also no notion of link bandwidth beyond the capacity of the wire for the transport of energy, which is more akin to channel capacity than link bandwidth in networking scenarios. Given these differences, only simulation tools specifically designed to model networking behaviours are considered in this survey.

1.2. Comparison to Other Surveys

Despite the recent provision of survey papers considering IoT and 6G communications, the forthcoming sections remark how these did not ponder on their suitability for simulating complex smart city scenarios as required by our aforementioned use case, as well as modelling State-Of-The-Art (SOTA) bottleneck mitigation and load balancing strategies.

1.2.1. IoT Simulators Survey

A survey of IoT simulators by Almutairi et al. [13] evaluated the current state of modern simulators that capture communications to and from IoT devices, addressing the research question of their ability to model arbitrary IoT communications. This remarks that no such simulator exists, with the lack of support for modelling renewable energy sources and security issues, such as battery attacks, being two of the most common omissions. SimulatorBridger (SB) [14] was one of the most comprehensive simulators, as it captures mobility in IoT and SDN architectures, energy models, and renewable energy sources, while being open-source and scalable (the metrics used to evaluate simulators within [13]). IoT simulators can overlap into other categories, e.g., SB is both an osmotic and IoT simulator, additionally this paper does not aim to repeat the work of Almutairi et al. and so does not re-compare all the simulators in the explicit category of IoT simulators; instead, as this paper has a different focus to that of [13], this paper only includes those most relevant to the discussion of smart city simulation.

1.2.2. 6G Simulators Surveys

Inzillo et al. [15] analysed the functionality of commonly used simulators and found that most did not support the simulation of technologies beyond Wi-Fi 5. They go on to say that the lack of simulators capable of keeping up with emerging technologies is a bottleneck for these technologies for commercial deployments in the real world. A more recent work by Evgenieva et al. [16] carried out a comprehensive survey of modern 6G simulators to determine the current state of such simulators to best “facilitate the accurate and effective modelling of future communication networks”. Evgenieva et al. grouped the simulators into three main categories: Link Level Simulators (LLSs), System Level Simulators (SLSs) and Network Level Simulators (NLSs). LLSs aim to accurately model the point-to-point connections between network loads, modelling signal fading, noise, interference, and propagation delay to determine their impact on transmission quality. LLSs also include channel simulators like [17], and aim to “evaluate the performance of different channel estimation and equalisation techniques”. SLSs, on the other hand, aim to “evaluate the performance of large-scale cellular networks”, while considering metrics such as network coverage, capacity, and energy efficiency [18]. These metrics are vital for optimising networks with “varying QoS requirements”. Different simulators focus on different simulated scenarios, and so QoS requirements can vary from packet loss [19], bit rate [20], and jitter [21] to overall throughput [22], latency [23] and general network availability. Finally, NLSs can model entire communication networks and simulate network behaviours spanning multiple network layers. NLSs are used to simulate how network protocols behave under varying conditions and how the network routes, switches and gateways interact when used in various network topologies. This enables the testing of routing algorithms, load balancing strategies, traffic management systems, and QoS requirements over large-scale systems, which is useful for optimising complex networks, including cellular, IoT, and satellite-based connectivity. NLS simulators clearly are the only type enabling the simulation of a smart city's network infrastructure as addressed in the present survey. Based on the findings of [16], there is only a single 6G-capable simulator within the NLS category: NYUSIM (NS) [24] (Section 3.3.2). Still, the 6G simulators were not evaluated against a specific goal, like in this paper, but were investigated to discover what the current state-of-the-art 6G simulators were capable of. Thus, the present survey provides an essential addition to the previous literature by providing a qualitative analysis through a comparative analysis.

1.3. Contributions

The purpose of this paper is to provide an overview of the current SOTA network simulators, cellular simulation tools and bottleneck mitigation strategies. Whilst there are many simulation tools capable of simulating various aspects of smart city infrastructure, most simulation tools are designed as single-purpose tools aiming to help answer a specific research inquiry. This limits the testing and development of new potential bottleneck mitigation strategies, as current SOTA strategies can only be tested within the confines of the utilised simulation environment. This paper first reviews the current SOTA network and cellular network simulation tools, while considering SOTA bottleneck mitigation and load balancing strategiesthe following desiderata:

ResQ №1 Which simulators are capable of supporting the simulation of entire communication infrastructures through the connection of IoT devices to the cloud through the edge?

ResQ №2 Which simulators are capable of simulating mobility agents and their interactions through Artificial Intelligence (AI) algorithms with the real world?

ResQ №3 Which simulators are ready to support future communication modelling through satellite communication and 6G architectures?

ResQ №4 Which simulators are able to model the utilisation of smart city resources?

ResQ №5 Which bottleneck mitigation techniques and load balancing strategies are most applicable to modern and future smart cities?

This survey's overall goal is to not only determine if any of the simulation tools are currently able to model a full smart city's network traffic with such bottleneck and load balancing strategies, but, failing that, if any are in a position to be improved to meet such functionality without fundamental structural overhauls.

The results from this survey indicate that there are currently no simulation platforms currently capable of fully modelling a full smart city infrastructure as per Section 1.1, with only SimulatorOrchestrator (SO) [5] positioned to model multiple sectors. However, even SO is currently only able to model vehicular and stationary health care IoT with no current support for mobile pedestrians' devices (e.g., mobile phones) or their use of public transport. Such support is required to model large city events such as concertsso as to better assess the impact of the subsequent increased network strain . Additionally, the SOTA bottleneck mitigation strategies and load balancing techniques outlined in this survey are not all currently implementable within the simulation tools. This is another key shortcoming of such simulation platforms, as it is insufficient to model, for example, the network traffic from a significant sporting event if the desired load balancing strategies cannot also be implemented within the same simulation environment.

1.4. Paper Organisation

The remaining sections of this paper outline the current state of modern network simulators and bottleneck mitigation strategies and evaluate their suitability for use in achieving a single federated simulation platform capable of modelling a full smart city's network infrastructure. Section 2 outlines the current state of mobility IoT simulators (Section 2.1), cloud simulators (Section 2.2), and osmotic simulators (Section 2.3), outlining their strengths and weaknesses derived from their respective focuses on a given area of the network infrastructure. Section 3 discusses simulators focusing on modelling communications to and from devices over current cellular networks (Section 3.1), but also looks ahead to future 6G technologies (Section 3.2) and simulators capable of modelling communications over likely 6G architectures (Section 3.3). Sections 4.1 and 4.2 analyse current SOTA bottleneck mitigation strategies and load balancing techniques along with their applicability for use within smart cities and smart city simulators. Section 5 provides a discussion on which of the current SOTA solutions best address the research questions outlined in Section 1.3. Finally, Section 6 summarises and concludes the paper with future suggestions on the direction smart city simulation could take to achieve the goal of a single simulator platform solution.

2. Network Simulators

Modern network simulators are essential in developing applications and (IoT) communication protocols [25], which are usually difficult to implement. By testing them with realistic data, we eliminate the need for realising this in the real world through costly hardware. The downside of utilising simulators in place of real-world experiments is that simulating and modelling many devices communicating over multiple communication types with intricate communication patterns is highly complex and computationally expensive [26]. The result of this complexity is that no modern simulator can fully simulate the seven Open Systems Interconnection (OSI) layers [13], instead focusing only on the layers most relevant to the simulator's intended use case. For certain use cases, particularly cybersecurity, the lack of all seven OSI layers is more detrimental, as there are attacks targeting each layer and defences that require various layers to work cooperatively to protect against vulnerabilities [27]. However, for simulating IoT devices within smart cities and the simulation of osmotic and SDN-based smart city infrastructure, the Presentation and Session layers are the most important. The simulation of these two layers enables an investigation into how different routing and scheduling algorithms, as well as various network topologies within the smart city infrastructure, affect the performance and behaviour of the network.

The papers outlined in this section were chosen as they were found to be the best suited to providing an overview of the current state of network simulators whilst still being relevant to simultaneously addressing the gap in current research along with ResQ №1–ResQ №3 outlined in Section 1. The chosen papers outline the various types of network simulators, and each is addressed, outlining their general strengths but also their weaknesses and limitations specific to full smart city simulation. Table 1 summarises our findings.

Table 1. Overview of the functionality of the simulators discussed within this section. A full circle represents a feature that is fully supported, and a half-circle represents partial support. Green indicates a feature that is beneficial to smart city infrastructure simulation, and red indicates a feature that is a detriment to smart city infrastructure simulation. A dash represents a feature that is absent from a given simulator.

Feature	**Simulator**							
	FC [28]	**CS7G** [29]	**S2C** [30]	**ISDWAN** [31]	**ISO** [32]	**ISOR** [26]	**SB** [14]	**SO** [5]
Cloud Processing	-	🟢	🟢	🟢	🟢	🟢	🟢	🟢
Cloud Workloads	-	🟢	🟢	-	-	-	-	-
Cloud Network Resource Management	-	🟢	-	🟢	🟢	🟢	🟢	🟢
Cloud Energy Resource Management	-	🟢	-	◖	◖	◖	◖	◖
Federator Device Selection	🔴	-	-	-	-	-	-	-
SD-WAN SDN Support	-	-	-	🟢	🟢	🟢	🟢	🟢
Edge Processing	◖	-	-	-	🟢	🟢	🟢	🟢
Edge Network Resource Management	◖	-	-	-	🟢	🟢	🟢	🟢
Edge Energy Resource Management	-	-	-	-	◖	◖	◖	◖
IoT Device Support	🟢	-	-	-	🟢	🟢	🟢	🟢
Mobility IoT Device Support	🟢	-	-	-	-	-	🟢	🟢
IoT Device Energy Modelling	-	-	-	-	◖	🟢	🟢	🟢
Osmotic Computing Support	-	-	-	-	🟢	🟢	🟢	🟢
Cellular Network Support	-	-	-	-	-	-	🟢	🟢
6G Infrastructure Support	-	-	-	-	-	-	-	🟢
AI Algorithm Support	-	-	-	-	-	-	-	🟢

2.1. *Mobility Simulators: FedCime (FC)*

Overview

To address the challenges of mobility within federated learning environments, Agabje et al. [28] propose FedCime (FC) to select actively the devices most likely to remain within the federator's coverage for the longest times. By analysing the expected delays of the transmissions from mobile devices, those with transmissions most likely to finish within the training time can be selected. Agbaje et al. expect that some of the chosen devices will migrate out of the training environment during training; to combat this, the federator chooses more devices than are minimally required for the training of the model, and if a device previously selected for training does leave, it is swapped out using one of the excess devices. A similarity score is given to the excess devices, and the device most similar to the training set is chosen from the excess set.

Pros

FedCime was validated through testing against other SOTA models, varying the migration rate of training devices and was successfully able to reduce the negative impacts of mobility within federated learning environments. The federator's ability to account for migrating devices by replacing them with equivalent replacements improved the accuracy and convergence of the models produced while also being more scalable than other models tested. Agbaje et al. tested both homogeneous and heterogeneous datasets, and FC outperformed the other models in both instances.

Cons

In terms of resource utilisation, FedCrime only mentions the computational overhead involved with aggregation of the weight values with no reference to energy or bandwidth utilisation. The key drawback of this approach, however, is that the results do not generalise to non-federated learning use cases, such as the simulation of smart cities' IoT infrastructure. In such environments, there is no federator to select which devices do and do not communicate. In most IoT environments, including smart cities, third-party devices communicate with IoT infrastructure with no overseer to directly control their exact behaviour. As such, the cloud bottlenecks outlined earlier would remain, as FC's solution to limit the number of devices allowed to transmit data would not be possible. To address such bottlenecks in general IoT environments, such as smart cities, additional mitigation strategies will be necessary. Therefore, FC would not be suitable as a framework for the simulation of smart city IoT infrastructure, and so simulating the behaviour of such IoT environments will require other tools to simulate the more generic behaviour of third-party devices transmitting communications according to each IoT device's, not a federator's, rule set.

2.2. Cloud Simulators

2.2.1. CloudSim (CS7G)

Overview

CloudSim (https://github.com/Cloudslab/cloudsim, accessed on 6 October 2025) is currently on its seventh iteration, named CloudSim 7G (CS7G) [29]. In this work, Andreoli et al. remark that the first iteration of CloudSim only modelled basic functionality of machine virtualisation and simple network behaviours. A key weakness of CloudSim 1.0 was the lack of a proper network model, a limitation which prevented the realistic simulation of big data and High Performance Computing (HPC) workloads. This led to NetworkCloudSim [33], which introduced a network flow model to account for the switches and physical machines inside a network. ContainerCloudSim [34] introduced the "evaluation of allocation, migration, and scheduling policies in containerised environments" through the addition of a new scheduling layer, which placed Cloudlets into containers which are then deployed into Virtual Machines (VMs). Andreoli et al. remark that not only is there a significant overlap between these External Modules, but that for each of these there is a single-feature extension of CloudSim proper. CS7G [29] coalesced these separate contributions to create a more cohesive framework whilst minimising needless repetition. with notable mentions being the provision of containers, geo-located services, web load balancing, and proper network flow modelling.

Pros

CloudSim 7G can now model the energy and computational costs of running workloads on VMs in cloud data centres. The primary goal of CS7G in [29] was to modernise CloudSim's architecture to allow for the easier integration of multiple External Modules on an as-needed basis, as well as improving the performance aspect of the codebase. Both of these were successful endeavours, with CS7G now able to evaluate "scheduling and resource management techniques for next-generation cloud computing environments" and utilising 25% less allocated memory than CloudSim 6G whilst finishing earlier in a set of controlled experiments.

Cons

Despite CS7G being the de facto standard in the evaluation of "resource management techniques in cloud computing environments,"it does not adequately support edge and IoT devices, such as simulating a smart city's network infrastructure. Despite IoT Sim-Edge [35] being one of such aforementioned extensions, which was within the added functionality, refined and integrated within [29] as the cloud remains the focus of CS7G, even though its functionality remains outside the scope of CS7G, as its main focus is on cloud simulation. This is further enforced within the future works of Andreoli et al., as they look to improve cloudlet scheduling accuracy for production-quality cloud environments as well as discuss adding components to CloudSim to allow the simulation of AI-enhanced resource management strategies in cloud computing.

2.2.2. Simcan2Cloud (S2C)

Overview

Maximising the efficiency of cloud data centres requires a thorough analysis of the data centres whilst processing various workloads, providing a cloud provider the necessary information on different resource allocation policies or how different workloads impact the monetary costs of a VM. Due to the complexities of using real-world production-ready cloud environments for such experiments, Canizares et al. [30] developed Simcan2Cloud (S2C), a discrete-event-based simulator for cloud environments, particularly aiming to address issues when simulating the infrastructure resources in cloud environments and the management and behaviour of users of cloud VMs–weaknesses Canizares et al. identified in other cloud simulators. To allow for the greatest comparison with real-world cloud systems, Canizares utilises real-world network traces collected from real production-ready systems. S2C is a C++-based simulation platform written on top off OMNeT++. In addition, ref. [30] also provides a GUI, allowing for the utilisation of S2C without requiring an in-depth knowledge of OMNeT++ and the NED language.

Pros

Simcan2Cloud was tested first with synthetic workloads, comparing multiple different CPU configurations, each with 10,000 users. The second test of S2C was using real-world network traces and Service Level Agreements (SLAs). This test aimed to investigate the impact of using different SLAs on the cloud's behaviour. They tested SLAs with no priority users or reserved machines, 10% priority users and 10% reserved machines, 30% priority user and 10% reserved machines and finally 30% priority users and 30% reserved machines. The results from these experiments demonstrated the success of the work of Canizares et al., demonstrating the capabilities of S2C as a simulation tool for modelling cloud-based systems. Given S2C's focus was on the cloud provider, it can calculate the monetary costs resulting from different SLAs, cloud deployments and VM configurations, as well as interactions between users and cloud services. S2C considers the physical resource requirements needed for the simulated tasks and even simulates the management process involved in allocating a suitable VM for a task before deployment.

Cons

However, S2C was only shown to be capable of realistically modelling cloud behaviours and has no support for IoT devices transmitting data to a cloud provider, with this being left for future work. As such, they also do not support mobility IoT devices. While users do not have a user-dedicated cloud VM, once a user has been assigned a VM for a task, that user remains connected to the VM until either the task is completed or the allocated time interval elapses. This means there is also no support for users dropping in and out of connection with a VM. The ability to simulate IoT devices, specifically mobile IoT devices, is necessary for simulating vehicles navigating a smart city. As the cars navigate the city, the attached IoT sensors are likely to lose and re-establish a network connection. Furthermore, S2C lacks functionality for modelling the energy resource utilisation involved in running the various workloads and SLAs within the cloud scenarios. The lack of all the above functionality prevents a tool like S2C from being utilised for a task like the simulation of a smart city's full IoT infrastructure.

2.2.3. IoTSim-SDWAN (ISDWAN)

Overview

Alwasel et al. [31] developed IoTSim-SDWAN (ISDWAN), able to simulate SD-WAN network environments with multiple SDN controlled cloud data centres. ISDWAN was developed to allow for the "modelling, simulating and evaluating of new algorithms" in "SD-WAN ecosystems and SDN-enabled multi-cloud datacentres". The general behaviour of transmitting a packet between two data centres within ISDWAN involves the generation of a packet at a source data centre host; the packet is then routed through the Software Defined Network Data Centre (SDN-DC), through the edge switch, the aggregate switch and the core switch before arriving at a gateway switch to the SD-WAN network. The packet is then routed through the SD-WAN network to the destination cloud data centre gateway switch, at which point it passes through the destination data centre's core switch, then an aggregate switch, and an edge switch before arriving at the destination data centre's host.

Pros

The accuracy of ISDWAN was validated through the comparison to real-world networks and then through application to a use case comparing WAN and SD-WAN topologies. Comparing SD-WAN and WAN topologies with respect to overall performance and energy consumption using ISDWAN, ref. [31] were able to show that the SD-WAN network was able to reduce transmission times by 50% compared to classical WAN topology, as the SD-WAN topology with the SD-WAN and SDN-DC controllers was able to dynamically locate the least congested parts within the network and route the network traffic accordingly. As for energy consumption, the SD-WAN topology consumed 54% less energy than the WAN topology, a result of the lower transmission times lowering the overall uptime of the network, lowering the energy consumption.

Cons

As with CS7G and S2C, the key drawback of this simulator is the focus on cloud data centres. The stated focus and all the demonstrated use cases within [31] are transmitting packets between different cloud data centres, and so there is no support within this work for communications between IoT devices, static or mobile, and cloud data centres. As stated within [31], user to data centre and data centre to data centre "require different management and policies", and so in order to simulate a smart city's network topology, communicating with mobile IoT devices would require extensions to a simulator such as ISDWAN.

2.3. Osmotic Simulators

Osmotic computing aims to improve the performance of IoT environments through the optimisation of various distributed systems which comprise the environment—these systems include edge computing, cloud computing and SD-WAN technologies [32]. Osmotic Simulators can provide a practical way to analyse various routing algorithms and QoS policies within osmotic environments without the need for real-world deployment. There are, however, challenges to simulating osmotic-IoT environments: maintaining coordination between the various types of distributed systems (e.g., edge with cloud); maintaining a consistent QoS with different message formats present within the subsystems (e.g., messages transmitted from the IoT layer to the edge layer may differ from the message type needed between the edge layer and the cloud layer); increasing complexity when adding more different layer types (e.g., an environment with just edge and cloud is less complex than the same environment with added IoT devices).

2.3.1. IoTSim-Osmosis (ISO) & IoTSim-Osmosis-RES (ISOR)

Overview

IoTSim-Osmosis (ISO) [32] aims to allow for the simulation of many osmotic systems in a single IoT environment. ISO supports IoT, edge, cloud and SD-WAN ecosystems and allows for the different components/systems to be provided with their own policies and routing algorithms. Within Alwasel et al. [32] outlines that ISO was designed to allow for the simulation of smart city-like environments, with dynamic management and IoT-oriented services across edge and cloud data centres via SD-WAN. Osmotic computing is essential in allowing various city domains to be connected in a more simplified manner and then managed through the deployment of MELs [36] across the edge-cloud resources. It then follows that osmotic simulators would then be important for developing effective management strategies. IoTSim-Osmosis-RES (ISOR), ref. [26], is an extension of ISO, ref. [32] adding support for Renewable Energy Sources (RES) as well as an 'osmotic agents module' for the simulation of autonomic IoT systems. The renewable energy support includes the simulation of data centres to be connected to photovoltaic power sources, including geographical data and the power output from the photovoltaics, as well as the capability to simulate energy storage to be used later. The 'osmotic agents module', on the other hand, assigns autonomous agents to IoT devices along with the edge and cloud data centres. The agents then autonomously implement a Monitor, Analyse, Plan, Execute (MAPE) loop [37]. This MAPE loop allows for the various aspects of the loop to be independently managed, with different rule sets or algorithms to be assigned to each stage to control an agent's behaviour. An AgentBroker component is responsible for the passing of messages between agents and executing the agents' MAPE loops.

Pros

Both ISOand ISOR were validated using case studies, both related to smart cities, for ISOR simulating data streams from IoT cameras streaming video to a central data centre. The cameras were powered by photovoltaic panels and were equipped with rechargeable batteries. ISOR was able to correctly show that the camera batteries contained less energy when there was less sun in the simulated environment, as well as showing the impact of different algorithms being used to control the behaviour of the autonomous osmotic agents.

Cons

ISOR and therefore ISO, do not support mobility IoT devices, only static IoT devices, which are vitally important for the simulation of smart cities and their connected infrastructure. Vehicles, and the transportation sector, are a large and important aspect of all cities, and IoT-enhanced vehicles (or AI-controlled vehicles) will not be able to be modelled with either ISOR or ISO.

2.3.2. SimulatorBridger (SB)

Overview

Vehicular Ad-hoc NeTworks (VANETs) consist of IoT-equipped vehicles and present additional challenges for simulation compared to Mobile Ad-hoc NeTworks (MANETs), as vehicles can enter and leave an environment during a simulation. To address these issues, Almutairi et al. [14] present SB, which combines the functionality of ISOR [26] with the SOTA traffic simulator Simulation of Urban MObility (SUMO) [38]. This combination utilises the mobility trace data within SUMO to generate Location Time-Step Pairs (LTSPs) for each vehicle equipped with an IoT sensor at each simulation time. These LTSPs are then utilised within ISOR to determine which IoT sensors are within the communication range of an edge node placed within the simulated urban environment. By utilising ISOR for the IoT network simulation SB retains the SDN/SD-WAN and osmotic agent support from ISOR whilst gaining the mobility support from SUMO.

Pros

SB was validated in [14], correctly mapping battery consumption to communication transmission in the simulated IoT city infrastructure, using realistic data on urban roads in a Bologna city environment. SB was expanded to allow for connection counting data [39] to be utilised in place of trace data, thus simply requiring the number of connections at each edge node and the time each communication is received at each edge node to recreate the overall network trends accurately.

Cons

SB is unable to model communications from a variety of different IoT protocols beyond Message Queuing Telemetry Transport (MQTT). Whilst the work of [39] would allow for the modelling of arbitrary IoT devices, the connection counting approach loses all IoT device data, making all communications identical. If a scenario required prioritising patient health data over vehicle data this would not be possible with connection counting data, and would not be possible to simulate at all with SB otherwise.

2.3.3. SimulatorOrchestrator (SO)

Overview

SO [5] further expands upon SB in several key ways. The first key approach is the extension to model 6G cell-free-based city infrastructure, as described in Section 3.2, along with the modelling of communication routing algorithms designed to leverage the improvements of such an infrastructure over cellular-based network infrastructure. The second key improvement, and the most important towards a single federated simulation tool, is the ability to run different simulators and digital twins in parallel, combining their respective network loads through the use of a Central Agent within a single simulated IoT infrastructure. This enabling the testing of the bottleneck mitigation and load balancing strategies implemented within SO to be tested with a network load generated from multiple data source types from different city sectors: the IoT-enhanced consumer vehicles from SUMO, like SB, and also multiple patient digital twins (which modelled patients with various health risk levels). These combined data sources transmitted their data to a cloud data centre through the same simulated smart city infrastructure. The Agentic Orchestrator's functionality was also extended [40] to control a fleet of ambulances in the simulation, where each separate vehicle was an Agentic AI. These, in turn, update the general traffic environment represented by SUMO. In fact, the ambulances were modelled as dynamic digital, thus extending then from passive and static digital twins to more proactive Agentic AI not only moving in the traffic scenario but also using environmental information and pre-trained information for re-pathing.

Pros

SO was shown to be capable of modelling network traffic incoming from two different domains, namely patient IoT devices and IoT-enhanced vehicles, demonstrating effective load balancing strategies to account for the combined network load. As per previous mention, the latest extension of such a simulator enabled the adoption of Agentic AIs orchestrated by a central AI-driven Agentic Orchestrator, thus adding a more dynamically adaptable version of the previous digital twins.

Cons

SO cannot adequately model the energy utilisation of the non-IoT aspects of the infrastructure. This is an important consideration for smart cities, and without the capability, it is difficult to fully assess the energy costs associated with any load-balancing mechanisms to compare with a cost-benefit analysis. It may not always be worth fully minimising communication transmission times if the associated energy costs are extremely high, and such considerations cannot be made without the inclusion of an infrastructure energy model.

3. Cellular Network Simulation

Early cellular networks enable high-speed mobile data transfer and were explicitly designed for Human-to-Human (H2H) communication applications, such as video chat, 4G cellular technologies, or LTE, thus making them an essential infrastructure to be considered as part of non-latency-sensitive smart city applications. Such applications include basic energy monitoring within smart cities [41] or traffic data collection within the urban transportation sector [42]. However, IoT devices generate network traffic with completely different characteristics to humans [43], which means it was not suitable for latency-sensitive applications such as real-time control of vehicle traffic. In addition to the improved latency compared to 4G, 5G also enables faster data transfer and has a higher network capacity, making it ideal for IoT applications within smart cities that support massive numbers of connected devices. The added 5G functionality allows for, in contrast to basic monitoring of smart city energy usage, real-time environmental monitoring, dynamic load balancing, and the integration of renewable energy into smart cities, allowing for smarter and greener energy usage [44].

High numbers of IoT devices, all collecting and streaming data over cellular networks within a smart city, will strain a network's resources, particularly bandwidth [45]; as such, network simulators are needed to test different network topologies with various load-balancing techniques and bottleneck mitigation strategies to find the optimal configurations for a given smart city's network infrastructure. It is therefore essential for such network simulators to accurately model the expected behaviour of the network when under a simulated network load. This subsection evaluates modern SOTA cellular network simulators, focusing on cellular network simulation scenarios, and highlights their strengths and weaknesses, while also identifying any research gaps addressed in this work.

Similarly to Section 2, the papers outlined in this survey aim to both address the gap in current research outlined in Section 1, whilst addressing ResQ №1–ResQ №3. The difference between the papers in Section 2 and this survey is that the focus of the papers in this section is on the modelling of communications over modern cellular networks (and beyond), and they are included to determine if a bottom-up approach is more suitable than a top-down approach (focusing on the infrastructure modelling). Table 2 summarises our findings.

Table 2. Overview of the functionality of the simulators discussed within this section. A full green circle represents a feature that is fully supported, and a half-circle represents partial support. A dash represents a feature that is absent from a given simulator. SB and SO are also included here, as they also support cellular network simulation.

Feature	**Simulator**							
	ROVS [46]	**NB-IoT [47]**	**CmM [15]**	**BC-I2 [17]**	**NS [24]**	**US [48]**	**SB [14]**	**SO [5]**
3G Support	-	-	-	-	-	-	●	●
4G Support	-	-	-	-	-	-	●	●
5G Support	●	-	-	●	●	-	●	●
Narrowband IoT (NB-IoT) Support	-	●	-	-	-	-	◐	◐
CF mMIMO Support	-	-	●	-	◐	-	-	●
Remote IoT Support	●	-	-	-	-	-	-	◐
Edge Support	◐	◐	●	-	-	◐	●	●
Cloud Support	-	-	-	-	-	-	●	●
Satellite Support	-	-	-	-	-	●	-	-

3.1. Modern Cellular Network Simulation

3.1.1. Quality of Experience (QoE) Testing with a Remote Operated Vehicle Simulator (ROVS)

Overview

The work of Schuler et al. [46] assesses QoE from the perspective of the remote operators when utilising 5G to connect with a Remote Operated Vehicle (ROV). The remote operators were asked to control the ROVs racing around a track through a video stream and rate the overall QoE. The overall QoE assessment was comprised of both qualitative (QoE) and quantitative (QoS) elements. The qualitative side was split into 4 aspects: video quality, artefacts, stuttering and usability. Remote operators manually evaluated this on a scale from 1 (low) to 5 (high), and the results were then averaged. The quantitative aspect of the experiment was determined through the lap times achieved by the remote operators under each of the latency and packet loss conditions. To evaluate the QoE and QoS [46] developed a "system-of-systems evaluation model" split into a vehicle component, an operator component and a transmission component the most relevant to the network simulation aspect. Schuler et al. routed the network traffic over both a network module and a channel module, which allowed them to control the constraints on the network (QoS) as they investigated how these constraints affected the QoE for the remote operators. The connection between the operator and the ROV was modelled as a direct link between the two and was designed to emulate the end-to-end nature of communication technologies.

Pros

The system of systems architecture of Schuler et al. was shown to be highly effective at demonstrating the necessary capabilities a real-world network required to operate remote vehicles over a video stream. The results do indicate that the absolute minimum latency and packet loss might be necessary to fully optimise the work of the remote operators, but that 99% of the performance can be retained with both additional latency, up to 80 ms on top of the 60 ms stream delay for their experiment, and packet loss, with packet loss values of 2.5% showing no quantitative signs of performance decline and even 1% having minimal reported impact by the remote operators. This granularity of result was only possible through the combination of qualitative QoE with quantitative QoS results, and is a real strength of their approach when dealing with any system involving human QoE. Additionally, their architecture's modular design allows for the various components to be swapped out. For example, the track racing performed by the remote operators can be swapped for any scenario or realistic simulation of any scenario involving remote vehicle operation. Furthermore, the network component can also be simply swapped out for a channel emulator, should a user want to use their architecture for Hardware in the Loop (HiL) scenarios.

Cons

The same authors remark that real-world remote operation requires the use of cellular networks "like 5G and beyond", however, in such real-world cellular networks the latency and packet loss have a combined dynamic figure, which they did not account for in their work. The energy utilisation of each configuration is also not considered as part of this work. Additionally, their architecture simulated only one connection between a single remote operator and a single vehicle, with the connection being upheld for the entire duration of the simulation. This does not allow a streamlined simulation of many IoT devices contemporaneously communicating and accessing the same infrastructure. Furthermore, thethe packet routing does not go beyond the first mile communication, while our urban scenario of interest requires the communication reaching the cloud (Section 1.1).

3.1.2. Stress Testing Narrowband IoT (NB-IoT)

Overview

The work by Jorke et al. [47] analyses the scaling limitations of the NB-IoT cellular network. NB-IoT is an IoT-based cellular network, specifically designed for battery-powered IoT devices, as it aims to minimise battery consumption through techniques such as Discontinuous Reception and minor data transmission optimisations. Additionally, NB-IoT includes improvements for the coverage of devices using the cellular network. NB-IoT was designed for small and efficient transmissions between IoT devices and first-mile antennae, enabling the batteries within the IoT devices to last for an extended period, while aiming to keep network latency below 10 s [47]. The work of Jorke et al. also compares two new transmission techniques, Cellular IoT Optimisation (C-IoT Opt) and Early Data Transmission (EDT), with a basic NB-IoT transmission, both of which aim to reduce the signalling overhead involved in the transmission of communications to lower network latency, power consumption and spectral usage.

Pros

The work is extensive and successfully demonstrated the scalability constraints of NB-IoT for three different transmission types. Their simulation supported over 1 million devices in outdoor, indoor, and deep indoor conditions, all transmitting data over the NB-IoT cellular network for 15 min, and then used these results to indicate expected outcomes over a full 24-hour period. Jorke et al. demonstrated how scaling affected the latency, the Packet Delivery Rate (PDR), and the impact of the number of Random Access Windows (RAWs) allocated within the uplink spectrum on the network once EDT removed the downlink spectrum utilisation bottleneck—a bottleneck they were able to identify thanks to their experiments and their simulation. Additionally, whilst not the focus of their work, the authors of the study state that their simulation tool can be utilised to model the energy consumption of the IoT devices communicating over the NB-IoT network.

Cons

A key drawback to their work is that they only simulated transmission to and from the first-mile antenna, while not considering the quintessential communication to the cloud as required by our scenario of interest (Section 1.1). Additionally, the IoT devices in this work are static within the cells, despite being placed in different locations. As such, Jorke et al. did not factor in how mobility IoT may impact the behaviour of the network as devices move between antennae, as would be the case with IoT-equipped vehicles within a smart city, as the vehicles drive around the city in and out of range of various first-mile antennae Furthermore, as the focus of the work within [47] was on the transmission of data from NB-IoT devices to first-mile antenna, there is no mention of the processing of the data once it reaches the antennae. This means that, even if the full infrastructure were supported, there would still be no support for determining the necessary Central Processing Unit (CPU) utilisation requirements to process the incoming communications once they reach the first mile antennae. There is little point in maximising the throughput of data to the antennae if the next step in the transmission of the data is bottlenecked by packet processing on the cloud, as all time savings will be lost. Thus, the packet limitation should not only consider the packet management on the first mile but also on the entire communication infrastructure.

3.2. Future 6G Architecture

With the number of connected devices ever increasing, coupled with a global push towards automation, current 5G technologies are unable to handle the required data transmissions [49]. Additionally, 6G aims to address these concerns with promises of improved performance, latency and availability. Also, 6G promises to make a reality many technologies which require "reliable, high bandwidth, and low latency connectivity".

A potential key technology for implementing 6G is Cell-Free massive Multi-Input Multi-Output (CF mMIMO). CF mMIMO distributes multiple Access Points (APs)/edge nodes throughout an environment to create a large area of uniform coverage suitable for many IoT devices [50]. This large uniform coverage area approach contrasts with the single cell tower per area approach of modern cellular networks. Cellular networks can suffer from issues such as inter-cell interference and decreased signal quality as a device moves further away from the cell tower [51]. CF mMIMO addresses these issues through the distributed nature of the edge nodes. Instead of each edge node being assigned a cell, the edge nodes work cooperatively in a decentralised manner, with no defined active region to provide the best signal to all connected devices. The resulting improvement to signal quality at all areas within the coverage region allows CF-mMIMO to be a suitable technology for future IoT networks [52]. One issue facing CF mMIMO networks is the scaling issue resulting from the high number of connected devices, which increases the computational complexity involved in handling their resulting network requests. The solution to this scaling issue comes in the form of User Centric Cell-Free massive Multi-Input Multi-Output (UC CF mMIMO), which limits IoT devices to connecting only to a small number of nearby cooperating edge nodes. This helps to minimise both the complexity required from the IoT device in choosing an edge node to connect to, as well as limiting the network traffic each edge node can receive [51]. The network traffic from the various edge nodes is then forwarded to a CPU [51,52].

In architectures with centralised computing, the transmission latency for devices at the edge can become a hurdle to fully utilising cloud computing. To avoid such issues, servers can be placed at the edge, allowing for the delegation of some of the cloud computing capabilities. This solution is referred to as Mobile Edge Computing (MEC) [53]. By placing MEC servers closer to the devices on the edge, transmissions can be pre-processed before forwarding communications to the cloud. MECs have been utilised within 5G technologies to meet “critical latency requirements of computationally intensive tasks”; combining MECs with UC CF mMIMO technologies will combine these improved edge latency benefits with the imported area-wide signal quality brought by the UC CF mMIMO. This combination is essential, as outlined in [53], “the success of edge computing [...] depends on the communication performance”.

When utilising the combination of MEC with UC CF mMIMO, it is possible to assign multiple edge nodes to a single MEC server; this requires that each MEC server be capable of managing each of their edge nodes. One solution to this is the use of SDNs, due to the ability of SDNs to control both the MEC servers’ resources and the associated edge nodes’ resources effectively through software [54]. The resulting SDN-controlled network topology closely resembles the osmotic network topology described in [26]. In both instances, there can be a central processing agent with perfect knowledge of the channel state information, information used to boost network performance [55] and manage and route transmissions from IoT devices transmitting communications towards a cloud data centre via edge nodes.

3.3. 6G Simulators

3.3.1. Cell-Free 6G mMIMO Simulator (CmM)

The work of Inzillo et al. [15] aims to update established network simulatorssuch as OMNeT++ to simulate both Wi-Fi 6 and CF mMIMO technologies in CmM, with a focus on understanding “the complex interplay among the physical layer attributes [...], data link layer mechanisms, and network layer routing protocols”. The nodes connected to a controller have communication radii which overlap to form virtual cells providing the uniform coverage areas outlined above. However, this validation only compares a theoretical throughput with the simulated throughput, with no comparison to real-world data, which was left for future work. Also left for future work was the addition of energy metrics, with there being no information within [15] for the energy usage of either the CF mMIMO infrastructure or any devices communicating with it.

3.3.2. Channel Simulators

Not all 6G simulators are focused on simulating the behaviour of the whole 6G infrastructure. Zhang et al. [17] focus on the changes to the channel characteristics and required antenna array designs for 6G communications, and present a simulator capable of modelling channels with such 6G requirements. They remark that 6G technologies will likely be targeting the terahertz (THz) frequency band, as 6G aims for a peak demand for 1 terabit per second (1 Tbps), in which the roughness of surfaces, transmission loses and molecular absorption become major challenges. As the number of connected devices rises MIMO technologies may also require an improvement, namely from massive MIMO to extreme MIMO technology "based on [...] much larger-scale antenna arrays". These new antenna arrays bring not only new improvements such as 100x the data rate but also shorter wave propagation distances, which result in new wave propagation characteristics which require investigation.

BUPTCMCCCMG-IMT2030 (BC-I2)

To investigate such new technologies, along with integrated sensing and communication (ISAC) and reconfigurable intelligent surface (RIS)-assisted channel simulation, and their resulting behaviours, Zhang et al. present BUPTCMCCCMG-IMT2030 (BC-I2), a simulation framework based on a 5G channel model, with the necessary extensions for the 6G technologies. The work of [17] is specifically focussed on the simulation of the channel behaviours resulting from 6G technologies, and as such would not be a suitable simulator to use to simulate a smart city with mobile IoT devices. The work of [17] aims to help investigate the necessary technology to make 6G a reality, whereas a simulator like the improved OMNeT++ for CF mMIMO [15] aims to show the benefits that using the 6G technology would bring. As outlined above, however, such simulators are not suitable for the modelling of smart cities' network infrastructures due to their focus on communications to and from first-mile antennae.

NYUSIM (NS)

NS [24] is "a critical tool for exploring wireless channels in the sub-THz spectrum". NS is able to support a wide variety of simulation environments, both rural and urban, including mMIMO-based environments, and is capable of producing realistic channel models suitable for real-world deployment. Whilst suitable for modelling channel behaviours in mMIMO-based environments, it is not designed specifically as an NLS to model the routing algorithms and load-balancing techniques required in such environments.

3.3.3. Satellite Technologies: UltraStar (US)

One of the aims of 6G is to improve and ensure better connectivity for all users irrespective of locationThis poses a challenge for users living in extremely rural areas [56], let alone those in extremely remote areas such as a desert or ocean [57]. The proposed solution to this problem is Satellite-Terrestrial Networks (STNs). They utilise Low Earth Orbit (LEO) satellites to help route data to and from users [58]. In order to meet the bandwidth requirements of a global demand with ever-increasing numbers of connected devices, large numbers of satellites will be required; such large numbers are referred to as "mega-constellations". Satellite-based networks with mega constellations are highly complex with dynamic topology and intermittent connectivity [48], with not only the movements of the satellites with respect to the earth affecting connectivity but also the satellites' movement with respect to other satellites.

Overview

Liu et al. [48] outline that current STN simulators are limited to small-scale scenarios and are both time-consuming and computationally intensive. UltraStar was then designed as a lightweight network simulator suitable for simulating mega constellations of connected satellites. UltraStar supports a large number of nodes, including the simulation of both ground and satellite network nodes. UltraStar was tested with 11,927 satellites positioned in the Walker constellation, an even and symmetrical distribution of satellites around the earth. They were able to investigate round-trip transmission time, multi-pathing routes for data from one point on earth to another, and link capacity.

Pros

The results from their experiments showed that not only were mega-constellations effective for use within communication networks, but also that UltraStar is an effective tool for simulating mega-constellations and is the first simulator to simulate over 10,000 satellites.

Cons

The key drawback to this work is the focus on satellite-based networks, and, as such, it is not suited to fully simulate our aforementioned urban scenario. Generally, for the simulation of vehicles navigating a smart city, satellite-based communication is not required, as satellites' primary usage is within rural and remote areas with limited connectivity, the precise opposite of the network conditions found within a highly connected smart city. Furthermore, there is no mention of the energy consumption involved in the transmission of the communications via satellite. One goal of this survey is to suggest that future communication infrastructures will utilise satellites within communication infrastructures, and the functionality of a simulator like UltraStar needs to be incorporated within future simulation tools aiming to model future smart cities' communication infrastructure.

4. Bottleneck Mitigation and Load Balancing Strategies

The papers in this section aim to address ResQ №5 and were chosen as these bottleneck mitigation and load balancing techniques are suited to relevant domains such as IoT scenarios and SDN/SD-WAN architectures. The techniques in this section were determined to be the current SOTA and therefore the solutions the desired simulators should be aiming to implement depending on the specific modelled scenario. Each of the included techniques have varying strengths and weaknesses, but all can play a part in helping smart cities avoid bottlenecks and meet high QoS demands. Table 3 summarises our findings.

Table 3. Overview of the functionality of the bottleneck mitigation and load balancing mechanisms discussed within this section. A full circle represents a feature that is fully supported, and a half-circle represents partial support. Green indicates a feature is beneficial to smart city bandwidth control, and red indicates a feature is a detriment to smart city infrastructure bandwidth control. A dash represents a feature that is absent from a given methodology and that this absence is generally neither good nor bad.

Feature	Algorithm								
	MQBS [59]	**MSBCS** [60]	**BFS** [61]	**WLB** [62]	**SPMB** [31]	**MCFR** [63]	**AADLB** [64]	**DRSTW** [10]	**PLA** [63]
Data Prioritisation Requirements	🔴	🔴	-	-	-	-	-	-	-
Data Prioritisation for Bandwidth	🟢	🟢	-	-	-	-	-	-	-
Bottleneck Detection	-	🟢	-	-	-	-	-	-	-
Bandwidth Prediction	-	🟢	🟢	-	-	-	-	🟢	-
Dynamic Bandwidth Reallocation	-	🟢	-	-	🟢½	🟢½	-	🟢½	-
Stops Working in Saturated Network	🔴	🔴	-	-	-	-	🔴½	-	-
Global Synchronisation Requirements	-	-	-	🔴		-	🔴	-	-
Can avoid optimal solution	-	-	-	-	🔴	-	-	🔴½	-
Requires IoT device data	-	-	-	-	-	🔴½	-	-	-
Results in data loss	🔴	🔴	-	-	-	-	-	-	-
Works in Stage 1 from Figure 1	-	-	-	-	-	-	-	-	🟢½
Works in Stage 2 from Figure 1	-	-	-	-	🟢½	🟢½	🟢½	🟢½	🟢½
Works in Stage 3 from Figure 1	🟢½	-	-	🟢	🟢½	🟢½	🟢½	🟢½	🟢½
Works in Stage 4 from Figure 1	🟢½	🟢	🟢	-	-	-	-	-	-

4.1. Bottleneck Mitigation Strategies

4.1.1. Multi-Queue Bandwidth Slicing (MQBS)

Overview

In SDN or SD-WAN-based IoT environments, data is transmitted from IoT devices to cloud data centres via edge nodes. According to the work of Habeeb et al. [59], however, within large-scale systems, such as smart cities, the following two common problems result from the heavy network traffic:

1. A transmission mismatch occurs when an edge node receives incoming transmissions more quickly than it can transmit, causing a bottleneck.
2. A processing mismatch occurs when an edge node cannot process its current transmissions quickly enough, which can again lead to a bottleneck.

The work of Habeeb et al. presents an IoT framework with a multi-level network-host queuing mechanism, data prioritisation and SDN network traffic slicing. Their research aims to minimise the overall transmission time of data from a set of IoT devices to the cloud while using as much bandwidth as possible at all times. Towards that aim, a multi-queue system, designed to reduce the queuing time of transmissions, dynamically separates incoming transmissions to an edge based on their priority and size. The next step is to slice the bandwidth needed for each transmission towards the cloud, once again based on its overall priority and size.

Pros

The solution improves the latency and throughput of data within the SDN/SD-WAN edge/cloud environments and overall improves QoS in the network. Not only were they able to reduce the processing times, the queuing times and the overall transmission times of data through the network, but they were, as a result, also able to reduce the energy consumption required by the network to transmit the data from IoT devices to the cloud, reducing the energy consumption by up to three times.

Cons

There are, however, some drawbacks. Firstly, both multi-queueing and bandwidth slicing require priority and the size of the data to function. If a network is sending transmissions of equal size and priority, then neither method will be applicable. Secondly, if the network is saturated, there will be little to no free bandwidth to divide among the queues, rendering bandwidth slicing ineffective. In such cases, additional mitigation strategies will be necessary to enhance the network's performance.

4.1.2. Multi-Stage Bandwidth Control Scheme (MSBCS)

Overview

The work of Omar Said, ref. [60], aims to develop a system to control the bandwidth and reduce the impacts of bottlenecks in IoT environments with many connected IoT devices.

Said's plan to address these issues is multi-stage. The first stage is a bottleneck detection mechanism. Due to IoT environments containing different types of devices and tools as well as different transmission techniques, Said deploys different techniques depending on if the network is WSN, RFID, MANET, or cellular.

Stage 2 consists of a Bandwidth Prediction approachexploiting machine learning techniques for traffic forecasting from historical data, specifically the Long Short-Term Memory (LSTM) algorithm. Said uses a neural network tasked with identifying the network type, the the type of data, the size of the data, the priority of the data, the complexity of the LSTM model, and the bandwidth utilisation.

Stage 3 reduces the bandwidth usage of the networkin three ways:decreasing the complexity of the process (such as decreasing QoS requirements), decreasing the number of transmitted packets, and reducing the size of packets transmitted. The goal for each of these three strategies is to reduce the number of bits transmitted through the network, reducing the required bandwidth. The philosophy behind each strategy is also the same: if a process or packet is of a lower priority when a bottleneck is detected, then either a smaller trimmed-back version is sent in place of the full process/packet, to save bandwidth, or failing that, the less important process/packet is simply not sent at all.

The final stage of Said's work is a bandwidth management model which considers both the real-time and predicted bandwidth problems in a network. The bandwidth reassignment plan uses information regarding the location of a bottleneck, the bandwidth of the surrounding areas and the importance of the data transmitting in the bottlenecked region. Next, an area in the surrounding region is chosen based on proximity to the bottlenecked area, the available bandwidth and the priority of the data transmitting. Bandwidth is then transferred from the selected region to the bottlenecked region. Next, a reallocation plan uses the same information regarding the location of the bottleneck and surrounding areas as the reassignment plan. This information helps to determine the frequency and severity of bottlenecks occurring in a particular region. If the problem is minor and requires little bandwidth, this bandwidth is, as before, taken from the surrounding areas. If, however, the issue is severe or persistent, an increase in bandwidth is requested from the Internet Service Provider (ISP) for that region.

Pros

The full multistage bottleneck prediction, detection, and mitigation approach proposed by Said was highly successful in achieving its goal of reducing the negative impacts of network bottlenecks in IoT environments. Said's approach was tested with various transmission types and different data sizes and priorities. It was able to reduce network delays, packet loss, energy consumption, and, most importantly, the occurrence of bottlenecks in the network. This multi-stage plan was designed as a long-term solution to address bottlenecks, thereby reducing their negative impacts across IoT environments.

Cons

Such an algorithm becomes naturally suboptimal at network saturation, i.e., when there is no more bandwidth to assign, leaving the reassignment plan no better than the base case of no plan. The reallocation plan can address this issue by either requesting bandwidth in advance from surrounding areas or requesting additional bandwidth from the ISP. The first option here is not a long-term solution, as outlined above; in cases of persistent bottlenecks, plan two is suggested. However, requesting additional bandwidth has a monetary cost. This may simply be the cost of paying an ISP for bandwidth, or this may require improved infrastructure to support the required bandwidth. Additionally, the predictive aspect would not be helpful if the prediction is that the network will be saturated, nor would knowledge of the bottleneck's location. In such circumstances, an improved routing algorithm may be preferred, especially if the alternative is expensive infrastructure improvements. Finally, the bandwidth reduction mechanism is highly reliant on either sending fewer packets (to reduce the complexity of the processes), sending fewer transmissions, or sending salient and smaller packets . However, there are instances for which data loss is not an acceptable trade-off for network performance. In the case of smart policing, any data loss could lead to lost evidence and unsolved crimes [65]; or, in the case of smart road monitoring, data loss could delay critical road repairs and increase the number of road accidents [66]. In such circumstances, the bandwidth reduction mechanism from [60] would not be applicable.

4.1.3. Bandwidth Forecast Service (BFS)

Overview

When mobile devices move between antenna/edge nodes, the stability of the network connection between the device and the antenna is often negatively affected. Orsini et al. [61] propose that connectivity forecasting can be used to address this problem. Bandwidth Forecast Service provides mobile devices both current and predicted knowledge of the network's bandwidth utilisation to enable IoT device optimisation of when and how to access the network. Given that the connectivity of each mobile device will be different, Orsini et al. determined that the connectivity forecasting must be done on the device itself. However, the training required for their predictive model does not need to be done on a mobile device. Orsini et al. opted for an adapted federated learning approach. The data from the mobile devices is aggregated on a more powerful device to train a model, which is then distributed to the mobile devices. This allowed the model to be optimised for the specific edge/cloud environment in which it will be deployed.

Pros

The Bandwidth Forecast Service was tested on a Java Android micro-service which was integrated into CloudAware [67]. The micro-service only activates when a forecast is requested to save energy. Additionally, more or less accurate forecasts can be requested to allow the devices to save more energy as necessary. Orsini et al. simulated 20 users using a mobile device in a constantly changing environment using realistic mobile usage data from the Lausanne Data Challenge Campaign (LDCC) [68]. They compared the XGBoost model [69] with three other models: a NAIVE model, which assumes the current state of the network will persist; Least Absolute Shrinkage and Selection Operator (LASSO) [70] and an autoregressive model (AR), which simply forecasts the bandwidth utilisation only considering the past data of the device itself. The work of Orsini et al. successfully developed the Bandwidth Forecast Service which, when the XGBoost model, was used, was able to forecast future bandwidth to a reasonable level of accuracy. In addition, the LASSO and AR models were still useful in circumstances in which high-accuracy predictions were not essential. In addition, combining the forecasts mode across a selection of models was shown to further boost the forecasts' accuracy even over the XGBoost model. Furthermore, Orsini et al. state that their Forecast Service is applicable to a wide range of scenarios in which mobile devices have limited bandwidth and could benefit from connectivity forecasting.

Cons

When novel network traffic patterns emerge, the predictive model will be unable to accurately predict the future bandwidth. This will result in the misallocation of bandwidth, a limited, sometimes scarce, resource, and will negatively impact the network, potentially resulting in bottlenecks. Finally, the bandwidth forecasting is focused on bottlenecks occurring on the mobile devices and does not account for any bottlenecks occurring within the communication infrastructure. If the transmissions from the mobile devices are forwarded towards an edge node, and then, after processing, forwarded again towards a VM on a cloud data centre, then the Bandwidth Forecast Service from [61] would not be able to address any potential bottlenecks between the edge and the cloud. Additional mitigation strategies or forecasting techniques would need to be deployed to address such bottlenecks.

4.2. Load Balancing Techniques

Load balancing within a network involves routing the communications through the network to make optimal use of the resources available to the network. By optimising the load balancing within a network, it is possible to minimise both overloading and underloading individual network nodes, boosting the performance of the overall network. The bottleneck mitigation strategies in the previous section primarily aim to deal with a bottleneck once it arises. However, through optimising the load balancing of traffic, the network bottlenecks can be removed or even prevented.

4.2.1. Weighted Load Balancing (WLB)

Overview

Within real-world scenarios involving edge nodes, such as smart cities, the hardware configurations of the edge nodes are dynamic. Hagiwara et al. [62] investigate if this causes a problem to modern popular load balancing techniques, which primarily assume uniformity across the hardware in all edge nodes in a network. They generate different network loads and compare their proposed weighted load-balancing technique with a software load-balancing technique called Maglev. The claim of Hagiwara et al. is that their weighted load-balancing approach more accurately accounts for the differences in performance or hardware configuration of each node when allocating packets, resulting in a reduction in the response times of the edge nodes.

Pros

The work outperforms load balancing approaches in cloud environments containing edge nodes with non-uniform hardware configurations. Compared to standard Maglev, ref. [62] was able to reduce average response times by 20% and maximum response times by 45%. Additionally, it is suitable for computationally intensive applications, particularly AI applications, by having shorter response times.

Cons

As the weights of the weighted load balancing approach are required to be updated with a high frequency, in real-world cloud environments with a natural delay dependant on the geographical distribution of the server, such information cannot be transmitted promptly. This could result in a loss of synchronisation between the nodes as some nodes will receive updates before others, and some nodes could even receive multiple updates before other nodes receive a single update. This loss of synchronisation will cause each load balancer to forward packets to different/wrong destinations, a problem similar to the consistency issue experienced by even load balancers outlined earlier. Additionally, some cloud environments maintain uniformity across the hardware configuration of the network nodes; in such cases, load-balancing techniques will be more suitable.

4.2.2. Shortest Path Maximum Bandwidth (SPMB)

Overview

Within their work in which Alwasel et al. [31], alongside presenting ISDWAN, outlined earlier, introduce a routing algorithm suitable for SD-WAN network architectures. Alwasel et al. present a modified classical Dijkstra algorithm [71] which is able to allow for a dynamic network graph whilst finding the shortest path between all the nodes in the network. Their algorithm, named Shortest Path Maximum Bandwidth (SPMB) is designed to simulate SD-WAN and SDN-DC networks, and finds the minimum number of traversing nodes between and source host and target host within the network, and selects the final route through the network based on which route has the maximum bandwidth in real-time.

Pros

The experimental comparison outlined in Section 2.2.3, for ISDWAN, was performed by comparing SPMB to the classical Dijkstra algorithm, and as outlined in Section 2.2.3, SPMB resulted in lowering both the transmission times and energy consumption of the network by around 50%. These improvements resulted from the SPMB algorithm being adapted to suit the SD-WAN network topologies, providing greater control over the network and allowing it to dynamically adjust the transmission routes based on the real-time behaviour of the network, such as avoiding bottlenecked areas.

Cons

The main drawback of this approach consists of delegating a shortest path algorithm to the allocation of the bandwidth within the communication network, thus not considering potential strategies for resource maximisation. Thus, this implicitly assumes that a shortest path algorithm could minimise packet delay by considering the edges in terms of available bandwidth. Furthermore, this algorithm did not enable the potential re-wiring of the allocated bandwidth under a fluctuating number of communicating devices, which might also enable resource maximisation by better re-distributing the allocated resources within the network.

4.2.3. Minimum Cost Flow Routing (MCFR)

Overview

The work from [63] considers latency minimisation as a cost minimisation problem while considering dynamic reallocation of the channel bandwidth, thus addressing the shortcomings of Minimum Cost Flow Routing (MCFR). The network latency is considered as directly proportional to the distance between two communicating graph nodes. The cost minimisation problem is then mapped to a multi-source and multi-target minimum-cost flow problem, where the flow is the number of unique IoT devices transmitting packets in the network. Within the modelled network MELs, ref. [36], communicate with each other if they are within each others' communication radius; IoT devices also communicate with the MELs if they are within a MEL's communication radius. The channel capacity between two MELs is capped at the minimum value that all active MELs can maximally communicate simultaneously. The channel cost between either two MELs or a MEL and an IoT device is proportional to the distance between them. Following these guidelines allows for the overall latency minimisation to be found whilst also considering channel capacities.

Pros

MCFR was used within SB to route communications incoming from IoT devices to edge nodes with MELs towards a cloud data centre. Communications were transmitted from the IoT devices over the 3G, 4G and 5G cellular network to determine which algorithm was optimal for communication routing under various network loads. MCFR was shown to outperform SPMB for the three cellular networks in a direct comparison and within SO [5] when applied to a cell-free based network infrastructure was also shown to outperform SPMB resulting in lower transmission times for communications from IoT devices routed through an SD-WAN/SDN based architecture.

Cons

When the two algorithms, SPMB and MCFR, were paired with a packet limiter, MCFR was only able to outperform SPMB for one of the three cellular networks tested (4G)–with equal performance between the two for 5G. Additionally, the connection counting methodology within [72], assumes that the IDs of the communicating IoT devices is not known by the algorithm, be that MCFR or SPMB. However, within [5], the MCFR algorithm required this information to outperform SPMB. Therefore, in situations where the connection counting methodology of [72] is used, SPMB is preferred in cell-free-based infrastructures.

4.2.4. Application-Aware Dynamic Load Balancing (AADLB)

Overview

The recent work of Petrovic et al. [64] aims to tackle the lack of adaptive load balancing mechanisms for SD-WAN-based architectures. Static load balancing techniques are unable to dynamically adjust traffic based on current network conditions in instances of dramatic changes in the state of incoming traffic, which leads to both congestion and suboptimal resource utilisation. Additionally, Petrovic et al. outline that allowing a load balancing mechanism to be aware of the applications which are currently being used will enable AADLB to dynamically "distinguish between different types of traffic" and prioritise critical applications over non-critical applications to maintain QoS in cases of heavy traffic. The algorithm of Petrovic et al., named Application-Aware Dynamic Load Balancing (AADLB) aims to optimise "routing decisions using real-time performance metrics dynamically". Such performance metrics include CPU, memory utilisation, latency, and packet loss. AADLB is tested in a network environment with three WAN-based networks connected via WireGuard VPN tunnels, allowing all three networks and their connected machines to communicate. One of the three networks contains a Central Administrative Hub, which manages and maintains the entire system, including load balancing across the other two WAN-based networks, named WAN-1 and WAN-2. These two WANs host application services, with WAN-1 being the primary network with low latency and high QoS requirements, and WAN-2 serving as a backup for redundancy and testing.

Pros

The results from the experiments within [64] show AADLB resulted in lower CPU utilisation after WAN-2 was shut down, whilst maintaining more stable and efficient CPU utilisation (CPU utilisation stability was improved by 76.66% compared with the other approaches) and also reducing latency by up to 2.58% in test scenario 2. The key benefit from the work of Petrovic et al., aside from these improvements, was the highlighting of a lack of other dynamic load balancing approaches for SD-WAN-based network architectures. Smart cities, with such an SD-WAN-based architecture, will also benefit from any advancements in this area.

Cons

The shortcomings of this approach are similar to Said [60]. If the network becomes overloaded, and each service provider becomes overloaded, then it will matter less and less which provider receives the latest network traffic. As stated earlier, AADLB never holds or drops any requests; it always allocates requests to the service application with the most resources. This can become a detriment in highly congested networks, as the application with the most resources could still have very few resources. Allowing for a floor to the acceptable available resources and holding requests if this threshold is not met could enable networks in a highly congested state to recover before continuing with the regular operation of AADLB, as currently outlined in [64].

4.2.5. Deep RNN SD-WAN Traffic Management (DRSTW)

Overview

When SD-WAN architectures receive incoming communications from IoT devices, graph search algorithms are used to route these communications towards cloud data centres. The running of these graph search algorithms takes time, which increases the network delay associated with the transmission of communications from IoT devices to cloud data centres. To reduce this search-related overhead, Absardi et al. [10] propose a deep learning approach that enables SD-WAN controllers to predict the network state and calculate a route using this information before the SD-WAN architecture actually receives the communications. Their model consists of a SD-WAN topology updater (which monitors the network topology for any changes), a Network State Information Collector (which collects information on the state of the network such as channel bandwidth), and a Learning Unit alongside a heuristic algorithm (SPMB [31] in the case of [10]) and previous network information to calculate the predicted routes. The learning unit utilises network state information to determine the optimal path for resource allocation.

Pros

Absardi et al. utilised ISO and compared their methodology to just SPMB [31]. Their proposed methodology was shown to reduce transmission delay of IoT communications through the SD-WAN network due to the ability to determine the best route in advance of the communications reaching the SD-WAN architecture. It was able to distribute the bandwidth more evenly across the open SD-WAN links. The prediction accuracy of the deep RNN was also shown to be high with a low Mean Absolute Error (MAE).

Cons

There were a few instances in which the proposed methodology predicted incorrect values, resulting in increased transmission times. There are also questions around the scalability of the approach. Absardi et al. outline that their approach scales with the number of data centres: as the number of data centres increases, so too does the training time, as well as the efficiency of the model. Absardi et al. remark that the number of collaborative data centres is usually limited, and so the "model's scalability is appropriate". Within smart cities, the number of collaborative data centres could rise significantly, and therefore, more work will be needed to determine if this approach is suitable for such scenarios.

4.2.6. Packet Limiter Algorithm (PLA)

Overview

SB [63] combines the aforementioned load balancing algorithms, Shortest Path Maximum Bandwidth (SPMB) and MCFR, with a packet limiter. The packet limiter algorithm (PLA) operates by only allowing each MEL on each edge node to have a certain number of active transmissions outgoing towards the cloud (e.g., 3). Once a MEL on an edge node had three non-terminated transmissions towards the cloud, all subsequent communications would be held in a buffer until an active transmission had finished, allowing for a new transmission to start from that MEL. The purpose of this limiter was to reduce the number of active communications overall in the network, thereby preventing the bandwidth available to each channel from plummeting too low and causing the entire network to bottleneck.

Pros

The PLA within [63] was shown not only to reduce the combined transmission times by 9–10%, but also helped to reduce the cloud processing times of the communications. The PLA helped slow the rate at which communications reached the cloud, allowing the cloud to process its current workload before new communications arrived.

Cons

The PLA as in [63] outlined that the choice of three as the limiter's threshold was arbitrary and had little impact on the performance of the algorithm. This suggests that there are no further gains to be found beyond the 9–10% improvements without refactoring.

This was found to be the case within [5], where the simulated CF mMIMO architecture was able to route communications to the cloud too quickly for the cloud's processing, even with the limiter active with the simulated architecture.

5. Discussion

Towards the aim of having a single simulation platform with which to model a full smart city and its connected network infrastructure, Table 4 shows that the osmotic simulators (ISO, ISOR, SB and SO) meet the most demands for smart city infrastructure and scenario simulation, especially ISOR, SB, and SO, and are therefore the best simulation platforms for modelling smart cities and their connected infrastructures. Of the osmotic simulators, SO is the most feature-complete and, therefore, the best placed for achieving this single simulation platform goal, improving and extending upon SB, the simulation platform with the second most complete feature set, only slightly rivalled by ISOR, ref. [26], which SB used as its base for simulating smart city IoT infrastructure.

Table 4. Comparing the functionality of the most suitable simulators discussed within this paper. Green circles indicate a feature that is present in a given simulator. A dash represents a feature that is absent from a given simulator.

Feature	**Simulator**										
	FC [28]	**CS7G [29]**	**S2C [30]**	**ISDWAN [31]**	**ISO [32]**	**ISOR [26]**	**CmM [15]**	**US [48]**	**NS [24]**	**SB [14]**	**SO [5]**
ResQ №1											
Cloud Processing	-	●	●	●	●	●	-	-	-	●	●
Edge Processing	●	-	-	●	●	●	-	●	-	●	●
IoT Devices	●	-	-	●	●	●	-	-	-	●	●
SDN Support				●	●	●				●	●
SD-WAN Support	-	-	-	●	●	●	-	-	-	●	●
Osmotic Computing	-	-	-	-	●	●				●	●
3G, 4G, 5G Support	-	-	-	-	-	-	●		●	●	●
ResQ №2											
Agents System	-	-	-	-	-	●	-	-	-	●	●
RES Support	-	-	-	-	-	●	-	-	-	●	●
Mobility IoT Devices	●	-	-	-	-	-	-	-	-	●	●
Connection Counting Support [39]	-	-	-	-	-	-	-	-	-	●	●
Real Time Data Injection	-	-	-	-	-	-	-	-	-	-	●
AI Enhanced Vehicular Routing [40]	-	-	-	-	-	-	-	-	-	-	●
ResQ №3											
CF mMIMO Support	-	-	-	-	-	-	●	-	●	-	●
Satellite Support	-	-	-	-	-	-	-	●	-	-	-

5.1. *ResQ №1*

With respect to ResQ №1, Table 4 shows that both SB and SO contain the required functionality to model the entire communication infrastructure of a smart city. Both can model cloud and edge processing of communications from IoT devices over cellular networks through osmotically controlled SDN/SD-WAN architectures, with the additional support for modelling different power management strategies within smart cities added in ISOR. Both SB and SO can model a variety of network infrastructures and topologies, allowing them to address the ResQ №1.

5.2. *ResQ №2*

For ResQ №2, SB and SO are also the two simulators with the widest-ranging functionality(Table 4). Whilst, like both SB and SO, ISOR does allow for the modelling of agent-controlled systems and renewable energy sources, such as IoT device batteries, it lacks the support for mobility IoT and connection-counting support found in both SB and SO. In particular, ISOR lacks complete support for dynamic agents, such as either Digital Twins or Agentic AI. However, SB also does not fully meet the full requirements outlined within ResQ №2, lacking support for the mobile IoT devices' interactions with AI algorithms. It is only through the extensions made within SO with the real-time data injection that this functionality was made possible. SO provides support through the real-time injection of data for the modelling of digital twins for ambulances and patients within smart cities [5], as well as AI-enhanced vehicular routing, using live information from SO to route IoT-enhanced ambulances through a busy city environment [40]. This functionality separates SO from all the other simulators in its ability to meet the simulator demands outlined in ResQ №2.

A General Framework for Orchestrating Simulators

This then calls for a deeper discussion concerning the feasibility of the overarching Use Case Scenario described in Section 1.1. Figure 2 proposes a generalization of the SO architecture, for basically supporting multiple different types of simulators: a Controller (tasks in orange), which mainly acts as a Earliest Deadline First scheduler through all the simulators communicatin through message passing, an Environment simulator, which models primarily the physical and non-cyber environment where all agents are residing (tasks in yellow), an Agentic Orchestrator and some Agentic AI/Digital Twins autonomously moving within the scenario of interest (tasks in green), and a set of Communication Netowork simulators (tasks in pink), potentially communicating between each other through the backbone and physical proximity information stored within the Controller. Furthermore, the environment also stores information about the nearest or best access points that an Agentic AI agent can use to initiate the communication process.

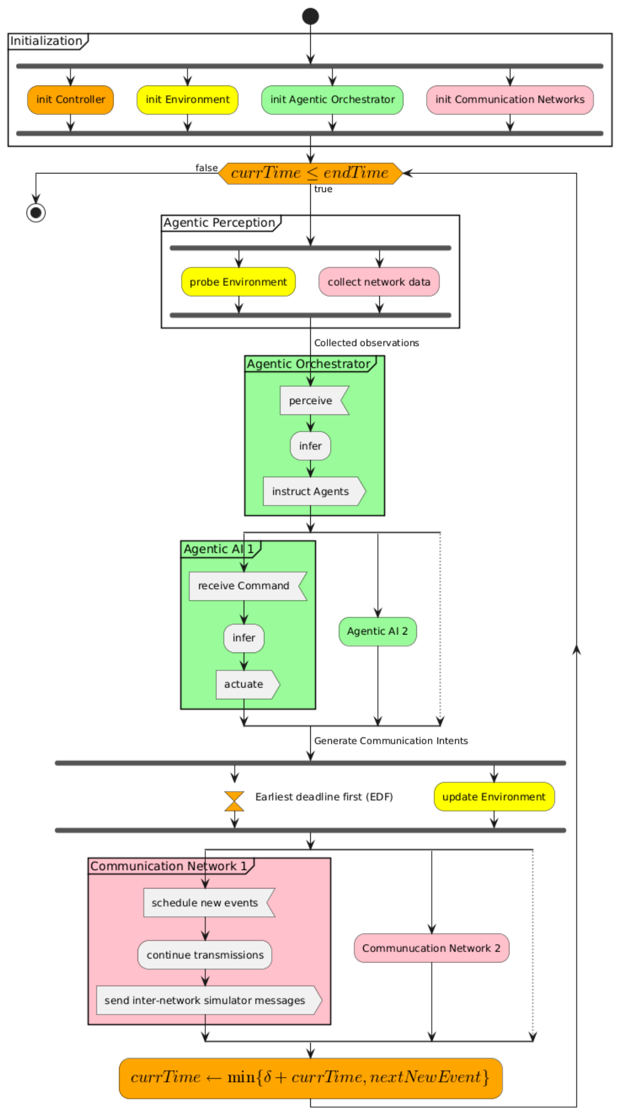

Figure 2. Activity Diagram [73] premiering a generic framework for orchestrating environmental and communication network perceptions, with their mental plan reasoning, as well as providing an outer

Earliest Deadline First scheduler for orchestrating communication events originated by both Agentic AI and Communication Networks towards other Communication Networks. Orange nodes represent the main Orchestrator events, green nodes refer to Agentic AI and Agentic Orchestration, green nodes represent the global Environment, while pink ones are the other simulators.

A generic Simulator Orchestrator integrating all the aforementioned components should work as follows: within the initialisation phase, the system loads the whole set of configurations, thus including the way the different communication infrastructures are wired together, therefore enabling the message passing from an Agentic AI and its nearest access point within a simulated Communication Network, as well as supporting communications occurring between different communication networks through explicit physical linking. The framework also initialises the physical position of the Agentic AI/Digital Twins in the Environment. At the time of writing, there is no system capable of orchestrating different possible communication infrastructures as a single entity, primarily because various authorities typically regulate these distinct communication networks; therefore, there is no need to support a holistic routing algorithm across these networks. Notwithstanding the former, the Controller is still required to support communications starting and ending at different simulators by enabling a communication strategy similar to Generic Routing Encapsulation [74]: while the final message that needs to reach destination is expressed as a message payload, the actual message dispatched to each communication network will only contain only the source and target node within the network, while the Controller will decide each time which is the best network towards which the packet should be dispatched in the following leg, until the message reaches destination and, therefore, it is entirely handled by the destination network without producing an outgoing message as a reply.

At each new incremental simulation step occurring at a given time *currTime*, we always check whether this value is less than the simulation time upper bound *endTime*. If this happens, we proceed with the simulation from the Agentic Perception; otherwise, we stop the simulation. To model both the Controller (or Central Agent [14]) and the Agentic AI within a similar framework, we divide the different stages into perception, inference, and action [75]. At each new simulation step, we require the physical (yellow) and cyber (pink) environments to send the central Agentic Orchestrator the most up-to-date information concerning the current state, as perceivable through the available sensors and statistics from the communication nodes. If the architecture contains an Agentic Orchestrator, it will then process the information in terms of actions to be executed by each chosen agent. If absent, each Agentic AI will receive the entire set of observations from the environment without any further mediation. Then, the agents might change their position in the physical world, thus requiring the physical environment to be updated (yellow). Contextually, the Simulation Controller will now collect all the communication intents from the agents, and sort the pending tasks using an optimal Earliest Deadline First algorithm [76].

At this stage, we can have two types of messages: either each Agentic AI initiates a communication, or a message traverses different simulated networks, as previously described, to reach its destination. Even if the communication starts and finishes within the same communication network of interest, each communication network could arbitrarily decide to offload some of its content to a secondary network for load-balancing purposes (e.g., Satellite Network [77]), or by letting the packet traverse multiple communication networks, thus requiring the usage of the Generic Routing Encapsulation approach described in the previous paragraph. Then, the Controller dispatches the message to the most appropriate network based on its internal routing information, which is derived from the initialised network infrastructure. We then end the simulation once the simulated time reaches the upper bound mentioned above.

Feasibility Study

Notwithstanding FC supporting the composition of multiple different communication networks, the proposed framework mainly works under the assumption that various communication networks are federated within a static configuration, for which the topology is mostly pre-set, primarily aims at connecting different IoT devices, thus not dealing with either dynamic Agentic AI, nor supporting complex interconnection between different communication infrastructures. Concerning the remaining simulators assessed within this survey, including SO, they all deal with a single component that simulates the entire communication infrastructure and do not consider a federation of interconnected simulators. Similarly, not all simulators exploit Agentic AI and Agentic Orchestration mechanisms to represent agents that dynamically change their behaviour contextually in response to updates on the communication infrastructure and the physical world, while only considering static agents with predetermined behaviour. Regardless, SO remains the simulator that implements most of the desired architecture, as it already implements a controller to manage messages coming from the agents through a scheduling algorithm, already manages the perception, inference, and actuation phases of the Agentic AI in the form of messages sent in the network and their change in physical position within an urban mobility scenario, and allows the simulation of an entire communication infrastructure through the extension of its previous version, SB.

As a final consideration, and given that each of these simulators roughly allows the simulation of only one single network, we are left wondering how easy it is to extend the revised and existing simulators so that they can be incorporated into the ecosystem presented in the figure above. As all the considered simulators, with the sole exception of UltraStar (US) and FC either come as an extension of a simulator framework or as an open-source software, it is always quite straightforward to extend existing simulators to let them simulate the transmission of new messages by creating one single common simulation interface as in [5], as they all simulate the dispatch of new messages. This is possible for all the simulators being extensions of CloudSim which, given that SO is an incremental work being the natural extension of ISDWAN, ISO, ISOR, and SB, are already included in SO. Given that S2C and Cell-Free 6G mMIMO simulator (CmM) are OMNET++-based simulators, those cannot be easily paused and resumed as the previous: the only possible solution would consist of adding clients accepting and sending the simulated messages through dedicated clients accepting connections from the outside (https://inet.omnetpp.org/docs/showcases/emulation/videostreaming/doc/index.html, accessed on 6 October 2025), while using other messages such as `alert` or `askYesNo` to temporarily pause the simulation after the simulated timeframe, for then resuming it through an external message. On the other hand, NS is Matlab-based, and therefore can be easily interconnected with the main framework if implemented in Python 3.12 (https://ch.mathworks.com/help/matlab/matlab_external/call-matlab-functions-from-python.html, accessed on 6 October 2025).

5.3. *ResQ №3*

For ResQ №3, Table 4 shows that none of the discussed simulators are capable of modelling both future 6G architectures and communications over satellites, with few simulators even capable of either. Only the works of Inzillo et al. [15], NS [24] and SO [5] are capable of modelling potential 6G mMIMO-based architectures, and only UltraStar [48] is capable of modelling satellite-based communications. This presents a gap in current research which future works will need to fill to accomplish the goal of modelling future communications architectures and environments such as smart cities.

5.4. *ResQ №4*

In terms of network resource utilisation, several of the network simulators outlined in Table 1 are capable of modelling both cloud and edge network resources and their management, including both SB and SO. However, a key aspect of smart cities is their push for sustainability, as such, smart city simulators need to be able to model not the energy usage within a smart city but specifically how this usage varies as a result of differing traffic policies, network architectures and load balancing strategies. We then introduce Table 5 for explicitly remarking which resources are tracked by each simulator.

Table 5. Comparing the resource monitoring functionality of the most suitable simulators discussed within this paper: while full circles remark complete support, half circles remark partial support. A dash represents a given simulator that does not support the given resource.

Resource Type Supported	**Simulator**										
	FC [28]	**CS7G [29]**	**S2C [30]**	**ISDWAN [31]**	**ISO [32]**	**ISOR [26]**	**CmM [15]**	**US [48]**	**NS [24]**	**SB [14]**	**SO [5]**
ResQ №4											
Energy Resources											
Cloud Energy Consumption	-	●	-	◐	◐	-	-	-	-	◐	◐
Edge Energy Consumption	-	-	-	◐	◐	-	-	-	-	◐	◐
IoT Device Battery	-	-	-	◐	●	-	-	-	-	●	●
Vehicle Battery	-	-	-	-	-	-	-	-	-	●	●
Power Management	-	●	-	-	-	●	-	-	-	●	●
Full Cloud Energy Modelling	-	●	-	-	-	-	-	-	-	-	-
Full Edge Energy Modelling	-	-	-	-	-	-	-	-	-	-	-
Full IoT Energy Modelling	-	-	-	-	-	●	-	-	-	●	●
Network Resources											
Simulated CPU Utilisation	-	●	●	◐	◐	◐	-	-	-	◐	◐
Simulated RAM / Memory Utilisation	-		●	◐	◐	◐	-	-	-	◐	◐
Channel Allocation	-	●	-	●	●	●	●	●	●	●	●
Link Bandwidth	-	●	-	●	●	●	●	●	●	●	●
Variable Packet Size	-	●	-	●	●	●	●	-	●	●	●

5.4.1. Energy Resource Utilisation

Table 5 outlines that very few of the simulators have support for energy modelling, with only CS7G, ISOR, SB, and SO fully representing any aspect of the desired simulated architecture (IoT-Edge-Cloud). CS7G is fully capable of such energy modelling but is limited to only cloud energy consumption, without considering an entire communication infrastructure. No surveyed approach considers energy modelling at the edge, with the edge's energy utilisation not fully encompassed in any of the simulators outlined in this paper. As an example of the issue, SO (and subsequently, through SB and ISOR to ISO as well as ISDWAN) does not actually model the work performed on the edge (or in the cloud), but rather calculates how long such work would take, thus considering time as a resource of interest. These simulators all build on each other and are based on an earlier version of CloudSim. As an example of their functionality, once a packet chunk reaches the edge, the end transmission time is updated without requiring any explicit data processing capabilities. This does not impact the simulation communications within a smart SDN SD-WAN infrastructure from a network perspective. Still, as there is no actual work happening at the edges or the cloud, the energy modelling within SO, for both the edge and cloud, are solely based on basic operating energies based solely on the length of time an edge, or the cloud, is active and does not account for the energy used actually to process and forward data. This functionality is a necessity for SO to become the go-to platform for smart city simulations, to actually model the time considered to process a task using the received packet as a function of the energy consumption [78], rather than just considering the time required to transmit the ongoing packet. The work of IoTSim-Edge [35] repurposed the functionality of modelling cloud behaviours from CloudSim 5G (see [29]) to simulate IoT devices communicating with the edge. A similar task could be accomplished with the new CS7G to model the more comprehensive feature set of CS7G to model edge energy behaviours.

Finally, ISOR, SB and SO can all model IoT device energies. Given that SB and SO can also model mobile IoT devices which can have a separate batteries to the IoT sensor batteries, both simulators need, and do have, the capability to model the battery depletion of IoT devices resulting from communication transmission, and the depletion of IoT-enhanced vehicles' batteries as a result of navigating a smart city environment - the latter functionality is achieved through SUMO as seen in [40].

5.4.2. Network Resource Utilisation

As can be seen in Table 5, the simulators outlined in this survey provide much better general support for network resource utilisation compared with energy resource utilisation, with only FC not supporting any resource modelling. Most of the outlined simulators support control over packet sizes and can monitor link bandwidths and channel allocations. However, only FC and S2C do not support these features, with UltraStar not supporting variable packet sizes. In terms of RAM and CPU utilisation monitoring, CS7G fully supports both functionalities, whereas S2C can only monitor CPU utilisation; ISDWAN, ISO, ISOR, SB and SO can estimate the usage of both. Still, as outlined in the prior section, these simulators do not actually simulate work done on the cloud or edge. This all makes CS7G the best suited for resource utilisation monitoring, but due to its focus on the cloud, for the simulated scenario desired in this work, one of its derivatives (ISDWAN, ISO, ISOR, SB, and SO), which support at least partial modelling of each of the network features listed in Table 1 would be best suited.

5.5. *ResQ №5*

SOTA bottleneck mitigation and load-balancing techniques surveyed in the present paper are not suitable for use within SO: novel strategies were developed in their place to help maintain network QoS and mitigate for bottlenecks within its simulation [5,63]. This is not unexpected: as stated within the work of Said [60], much of the research into bandwidth management is either special purpose or outright not suited to IoT environments. However, the load balancing and bottleneck mitigation techniques outlined in this paper apply to specific IoT scenarios, but not useful for the scenario described in Section 1.1. Both the works of [60,62] require data prioritisation for their mitigation strategies, which does not apply to SO; communications to and from IoT devices and the cloud are identical, and so there is no basis on which to prioritise one communication over another. This uniformity also extends to individual packets, as [60] also suggests only transmitting the most important packets if a bottleneck is detected; this again does not apply to SO.

Additionally, as outlined in Section 4.1, data loss is not always an acceptable trade-off for network performance, and the mitigation strategies outlined in [5] do not result in any data loss. On the other hand, both SPMB and MCFR have already been implemented within SO, and as the RNN approach of [10] was tested with ISO this methodology would be suitable for implementation within SO. Whilst few of the mitigation strategies outlined in this paper were suitable for SO, the PLA outlined in [63] makes use of an idea similar to that of both [60,61]–mitigating bottlenecks through changing the delay between successive packets in transmissions from the edge nodes to the cloud.

None of the bottleneck mitigation strategies or load balancing techniques from Table 3 truly operated in the cloud data centres (Stage 1 from Figure 1), highlighting a gap in the current state of the art. The PLA had a secondary effect, which improved performance in the cloud; however, the PLA was implemented on the edge to help boost transmission times through the network. According to the results of [5], the cloud was identified as a source of network bottlenecks, even with the PLA, due to a CF mMIMO architecture paired with MCFR routing, which resulted in communications to the cloud being processed more quickly than the cloud could handle its current load. The suggestion to address this bottleneck, as presented in [5], was to implement a new PLA-like algorithm on the cloud, specifically to limit the rate at which communications started processing. This suggests that for future architectures paired with current SOTA bottleneck mitigation, the cloud may become the primary source of network bottlenecks due to the combination of the current SOTA addressing bottlenecks elsewhere and the lack of cloud-specific bottleneck mitigation strategies.

6. Future Works

This survey premieres the definition of a whole-embracing smart city urban scenario (Section 1.1), which is then used to assess the existing communication simulators with respect to their ability to fully implement the communication architecture in Figure 1. Full compliance with these desiderata requires interconnection between different communication networks, which are presently supported by disparate simulators. Thus, this postulates a generic architecture that enables the interpretation of each simulator (Figure 2). Despite the technical difficulties for some, it should be potentially possible to integrate any simulator within this "meta-simulator". As different simulators also support the simulation of various types of IoT devices, this will enable support for disparate types of IoT devices. As a result of these considerations, our paper then focuses on discussing which of the aforementioned simulators can fully support dynamic agents that not only exploit the communication infrastructure but also interact with a physical world. Digital twins are, in fact, a vital tool for treating patient health conditions [79], especially in smart cities, and therefore need to be considered for comprehensive smart city simulation. On the other hand, modelling a fleet of cars dynamically repathing their route within an urban scenario requires the adoption of an Agentic Orchestrator, as well as Agentic AI agents that not only communicate but also perform actions directly on the physical environment by altering their physical state and presence. The present survey remarks how existing communication simulators are incapable of supporting this, with the sole exception of SO. Notwithstanding the former, the complete simulation of agents within an urban scenario requires modelling crowds and their interaction with public means of transport, which is currently unsupported. However, the availability of public transport is likely to have an impact on the number of pedestrians. Definitely, their movements in a given area mean that a public transport model would either need to interact with a pedestrian simulator in advance of it injecting data into SO, or a SOTA model, which already models both pedestrians and public transport in a given area, needs to be utilised. Future works will consider modelling this interaction in SUMO, thus testing whether this tool can be fully exploited as an environment simulator, or whether a new environment simulator is required to model more complicated mobility scenarios.

Future works will require the design of a new simulator more adherent to the proposed "meta-simulator" framework. In fact, despite SO's ability to implement most of the desiderata from Figure 2, there are crucial extensions that still require major reworks. SO presents the first implementation of this architecture, where the controller orchestrates between the agentic orchestrator, an entire edge/cloud communication infrastructure via SB, and environment data dynamically generated by the VANET SUMO simulator. As the SO 's controller is already implemented as an EDF scheduler, minor work needs to be carried out to scale the proposed architecture to meet the desiderata from Figure 2. Still, given that SO was working on an older extension of CloudSim via ISO and ISOR, this cannot adequately support work processed on both the edge and on the cloud, as now supported by CS7G. Future works will also investigate the possibility of further extending Figure 2 for considering simulator modularity and composition through more complex orchestration interactions [80]. In fact, CS7G capabilities do not consider data prioritisation mechanisms as found in real network systems. Not only would this increase the realism of the simulated scenarios capable with SO, but it would also allow for the implementation of bottleneck detection and mitigation strategies like those in [60,62], which are currently suitable for real smart cities.

Author Contributions: Conceptualization, methodology, formal analysis, R.G.; writing—review and editing, G.B.; writing—original draft preparation, R.G.; supervision, project administration, G.B.; funding acquisition, G.B. and G.M.; investigation, resources, G.B. and G.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AADLB	Application-Aware Dynamic Load Balancing
AP	Access Point
BC-I2	BUPTCMCCCMG-IMT2030
BFS	Bandwidth Forecast Service
C-IoT	Opt Cellular IoT Optimisation
CF	mMIMO Cell-Free massive Multi-Input Multi-Output
CmM	Cell-Free 6G mMIMO simulator
CPU	Central Processing Unit
CS7G	CloudSim 7G
DRSTW	Deep RNN SD-WAN Traffic Management
EDT	Early Data Transmission
FC	FedCime
H2H	Human-to-Human
HiL	Hardware in the Loop
HPC	High Performance Computing
IoT	Internet of Things
ISDWAN	IoTSim-SDWAN
ISO	IoTSim-Osmosis
ISOR	IoTSim-Osmosis-RES
ISP	Internet Service Provider
LASSO	Least Absolute Shrinkage and Selection Operator
LDCC	Lausanne Data Challenge Campaign
LEO	Low Earth Orbit
LLS	Link Level Simulator
LSTM	Long Short-Term Memory
LTSP	Location Time-Step Pair
MAE	Mean Absolute Error
MANET	Mobile Ad-hoc NeTwork
MAPE	Monitor, Analyse, Plan, Execute
MCFR	Minimum Cost Flow Routing
MEC	Mobile Edge Computing
MQBS	Multi-Queue Bandwidth Slicing
MQTT	Message Queuing Telemetry Transport
MSBCS	Multi-Stage Bandwidth Control Scheme

NB-IoT Narrowband IoT	
NLS	Network Level Simulator
NS	NYUSIM
OSI	Open Systems Interconnection
PDR	Packet Delivery Rate
PLA	Packet Limiter Algorithm
QoE	Quality of Experience
QoS	Quality of Service
RAW	Random Access Window
RES	Renewable Energy Sources
ROV	Remote Operated Vehicle
ROVS	Remote Operated Vehicle Simulator
S2C	Simcan2Cloud
SB	SimulatorBridger
SD-WAN	Software-Defined Wide Area Network
SDN	Software Defined Network
SDN-DC	Software Defined Network Data Centre
SLA	Service Level Agreement
SLS	System Level Simulator
SO	SimulatorOrchestrator
SOTA	State-Of-The-Art
SPMB	Shortest Path Maximum Bandwidth
STN	Satellite-Terrestrial Network
SUMO	Simulation of Urban MObility
UC	CF mMIMO User Centric Cell-Free massive Multi-Input Multi-Output
US	UltraStar
VANET	Vehicular Ad-hoc NeTwork
VM	Virtual Machine
WLB	Weighted Load Balancing

References

1. Syed, A.S.; Sierra-Sosa, D.; Kumar, A.; Elmaghraby, A. IoT in Smart Cities: A Survey of Technologies, Practices and Challenges. *Smart Cities* **2021**, *4*, 429–475. https://doi.org/10.3390/smartcities4020024.
2. Grishina, A.; Chinnici, M.; Kor, A.L.; Rondeau, E.; Georges, J.P.; De Chiara, D., Data Center for Smart Cities: Energy and Sustainability Issue. In *Big Data Platforms and Applications: Case Studies, Methods, Techniques, and Performance Evaluation*; Pop, F., Neagu, G., Eds.; Springer International Publishing: Cham, Switzerland, 2021; pp. 1–36. https://doi.org/10.1007/978-3-030-38836-2_1.
3. Jawhar, I.; Mohamed, N.; Al-Jaroodi, J. Networking architectures and protocols for smart city systems. *J. Internet Serv. Appl.* **2018**, *9*, 26. https://doi.org/10.1186/s13174-018-0097-0.
4. Weingartner, E.; vom Lehn, H.; Wehrle, K. A Performance Comparison of Recent Network Simulators. In Proceedings of the 2009 IEEE International Conference on Communications, Dresden, Germany, 14–18 June 2009; pp. 1–5. https://doi.org/10.1109/ICC.2009.5198657.
5. Gillgallon, R.; Almutairi, R.; Bergami, G.; Morgan, G. SimulatorOrchestrator: A 6G-Ready Simulator for the Cell-Free/Osmotic Infrastructure. *Sensors* **2025**, *25*, 1591. https://doi.org/10.3390/s25051591.
6. Bergami, G.; Packer, E.; Scott, K.; Del Din, S. Predicting Dyskinetic Events Through Verified Multivariate Time Series Classification. In Proceedings of the Database Engineered Applications, Bayonne, France, 26–29 August 2024; Chbeir, R., Ilarri, S., Manolopoulos, Y., Revesz, P.Z., Bernardino, J., Leung, C.K., Eds.; Cham, Switzerland, 2025; pp. 49–62.
7. Meliá, S.; Nasabeh, S.; Luján-Mora, S.; Cachero, C. MoSIoT: Modeling and Simulating IoT Healthcare-Monitoring Systems for People with Disabilities. *Int. J. Environ. Res. Public Health* **2021**, *18*, 6357. https://doi.org/10.3390/ijerph18126357.
8. Hussein, T.D.H.; Frikha, M.; Ahmed, S.; Rahebi, J. Ambulance Vehicle Routing in Smart Cities Using Artificial Neural Network. In Proceedings of the 2022 6th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP), Sfax, Tunisia, 24–27 May 2022; pp. 1–6. https://doi.org/10.1109/ATSIP55956.2022.9805857.
9. Sutherland, M.; Chakrabortty, R.K. An optimal ambulance routing model using simulation based on patient medical severity. *Healthc. Anal.* **2023**, *4*, 100256. https://doi.org/10.1016/j.health.2023.100256.

10. Nazemi Absardi, Z.; Javidan, R. A predictive SD-WAN traffic management method for IoT networks in multi-datacenters using deep RNN. *IET Commun.* **2024**, *18*, 1151–1165. https://doi.org/10.1049/cmu2.12810.
11. Jia, X.; Xia, Y.; Yan, Z.; Gao, H.; Qiu, D.; Guerrero, J.M.; Li, Z. Coordinated operation of multi-energy microgrids considering green hydrogen and congestion management via a safe policy learning approach. *Appl. Energy* **2025**, *401*, 126611. https://doi.org/10.1016/j.apenergy.2025.126611.
12. Zhang, S.; Gu, W.; Yao, S.; Lu, S.; Zhou, S.; Wu, Z. Partitional Decoupling Method for Fast Calculation of Energy Flow in a Large-Scale Heat and Electricity Integrated Energy System. *IEEE Trans. Sustain. Energy* **2021**, *12*, 501–513. https://doi.org/10.1109/TSTE.2020.3008189.
13. Almutairi, R.; Bergami, G.; Morgan, G. Advancements and Challenges in IoT Simulators: A Comprehensive Review. *Sensors* **2024**, *24*, 1511. https://doi.org/10.3390/s24051511.
14. Almutairi, R.; Bergami, G.; Morgan, G.; Gillgallon, R. Platform for Energy Efficiency Monitoring Electrical Vehicle in Real World Traffic Simulation. In Proceedings of the CBI, Prague, Czech Republic, 21–23 June 2023; pp. 1–8. https://doi.org/10.1109/CBI58679.2023.10187450.
15. Inzillo, V.; Quintana, A.A. Implementation of 802.11ax and cell-free massive MIMO scenario for 6G wireless network analysis extending OMNeT++ simulator. *SIMULATION* **2025**, *101*, 117–143. https://doi.org/10.1177/00375497241266256.
16. Evgenieva, E.; Vlahov, A.; Ivanov, A.; Poulkov, V.; Manolova, A. A Comprehensive Survey of 6G Simulators: Comparison, Integration, and Future Directions. *Electronics* **2025**, *14*, 3313. https://doi.org/10.3390/electronics14163313.
17. Zhang, J.; Lin, J.; Tang, P.; Zhang, Y.; Xu, H.; Gao, T.; Miao, H.; Chai, Z.; Zhou, Z.; Li, Y.; et al. Channel Measurement, Modeling, and Simulation for 6G: A Survey and Tutorial. *arXiv* **2025**, arXiv:2305.16616.
18. Kwon, S. Ensuring renewable energy utilization with quality of service guarantee for energy-efficient data center operations. *Appl. Energy* **2020**, *276*, 115424. https://doi.org/10.1016/j.apenergy.2020.115424.
19. Roy, A.; Acharya, T.; DasBit, S. Quality of service in delay tolerant networks: A survey. *Comput. Netw.* **2018**, *130*, 121–133. https://doi.org/10.1016/j.comnet.2017.11.010.
20. Dimitriou, N.; Tafazolli, R.; Sfikas, G. Quality of service for multimedia CDMA. *IEEE Commun. Mag.* **2000**, *38*, 88–94. https://doi.org/10.1109/35.852036.
21. Paulsen, S.; Uhl, T.; Nowicki, K. Influence of the jitter buffer on the quality of service VoIP. In Proceedings of the 2011 3rd International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), Budapest, Hungary, 5–7 October 2011; pp. 1–5.
22. Zhu, H.; Li, M.; Chlamtac, I.; Prabhakaran, B. A survey of quality of service in IEEE 802.11 networks. *IEEE Wirel. Commun.* **2004**, *11*, 6–14. https://doi.org/10.1109/MWC.2004.1325887.
23. Ademaj, F.; Bernhard, H.P. Quality-of-Service-Based Minimal Latency Routing for Wireless Networks. *IEEE Trans. Ind. Inform.* **2022**, *18*, 1811–1822. https://doi.org/10.1109/TII.2021.3071596.
24. Poddar, H.; Ju, S.; Shakya, D.; Rappaport, T.S. A Tutorial on NYUSIM: Sub-Terahertz and Millimeter-Wave Channel Simulator for 5G, 6G, and Beyond. *IEEE Commun. Surv. Tutor.* **2024**, *26*, 824–857. https://doi.org/10.1109/COMST.2023.3344671.
25. Gallegos Ramonet, A.; Pecorella, T.; Picano, B.; Kinoshita, K. Perspectives on IoT-oriented network simulation systems. *Comput. Netw.* **2024**, *253*, 110749. https://doi.org/10.1016/j.comnet.2024.110749.
26. Szydlo, T.; Szabala, A.; Kordiumov, N.; Siuzdak, K.; Wolski, L.; Alwasel, K.; Habeeb, F.; Ranjan, R. IoTSim-Osmosis-RES: Towards autonomic renewable energy-aware osmotic computing. *Softw. Pract. Exp.* **2022**, *52*, 1698–1716. https://doi.org/10.1002/spe.3084.
27. Sullivan, S.; Brighente, A.; Kumar, S.A.P.; Conti, M. 5G Security Challenges and Solutions: A Review by OSI Layers. *IEEE Access* **2021**, *9*, 116294–116314. https://doi.org/10.1109/ACCESS.2021.3105396.
28. Agbaje, P.; Anjum, A.; Talukder, Z.; Islam, M.; Nwafor, E.; Olufowobi, H. FedCime: An Efficient Federated Learning Approach For Clients in Mobile Edge Computing. In Proceedings of the 2023 IEEE International Conference on Edge Computing and Communications (EDGE), Chicago, IL, USA, 2–8 July 2023; pp. 215–220. https://doi.org/10.1109/EDGE60047.2023.00042.
29. Andreoli, R.; Zhao, J.; Cucinotta, T.; Buyya, R. CloudSim 7G: An Integrated Toolkit for Modeling and Simulation of Future Generation Cloud Computing Environments. *Softw. Pract. Exp.* **2024**, *55*, 1041–1058. https://doi.org/10.1002/spe.3413.
30. Cañizares, P.C.; Núñez, A.; Bernal, A.; Cambronero, M.E.; Barker, A. Simcan2Cloud: A Discrete-Event-Based Simulator for Modelling and Simulating Cloud Computing Infrastructures. *J. Cloud Comput.* **2023**, *12*, 133. https://doi.org/10.1186/s13677-023-00511-w.
31. Alwasel, K.; Jha, D.N.; Hernandez, E.; Puthal, D.; Barika, M.; Varghese, B.; Garg, S.K.; James, P.; Zomaya, A.; Morgan, G.; et al. IoTSim-SDWAN: A simulation framework for interconnecting distributed datacenters over Software-Defined Wide Area Network (SD-WAN). *J. Parallel Distrib. Comput.* **2020**, *143*, 17–35. https://doi.org/10.1016/j.jpdc.2020.04.006.
32. Alwasel, K.; Jha, D.N.; Habeeb, F.; Demirbaga, U.; Rana, O.; Baker, T.; Dustdar, S.; Villari, M.; James, P.; Solaiman, E.; et al. IoTSim-Osmosis: A framework for modeling and simulating IoT applications over an edge-cloud continuum. *J. Syst. Archit.* **2021**, *116*, 101956. https://doi.org/10.1016/j.sysarc.2020.101956.

33. Garg, S.K.; Buyya, R. NetworkCloudSim: Modelling Parallel Applications in Cloud Simulations. In Proceedings of the 2011 Fourth IEEE International Conference on Utility and Cloud Computing, Melbourne, VIC, Australia, 5–8 December 2011; pp. 105–113. https://doi.org/10.1109/UCC.2011.24.
34. Piraghaj, S.F.; Dastjerdi, A.V.; Calheiros, R.N.; Buyya, R. ContainerCloudSim: An environment for modeling and simulation of containers in cloud data centers. *Softw. Pract. Exp.* **2017**, *47*, 505–521. https://doi.org/10.1002/spe.2422.
35. Jha, D.N.; Alwasel, K.; Alshoshan, A.; Huang, X.; Naha, R.K.; Battula, S.K.; Garg, S.; Puthal, D.; James, P.; Zomaya, A.; et al. IoTSim-Edge: A simulation framework for modeling the behavior of Internet of Things and edge computing environments. *Softw. Pract. Exp.* **2020**, *50*, 844–867. https://doi.org/10.1002/spe.2787.
36. Carnevale, L.; Celesti, A.; Galletta, A.; Dustdar, S.; Villari, M. From the Cloud to Edge and IoT: a Smart Orchestration Architecture for Enabling Osmotic Computing. In Proceedings of the 2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA), Krakow, Poland, 16–18 May 2018. https://doi.org/10.1109/WAINA.2018.00122.
37. Gheibi, O.; Weyns, D.; Quin, F. Applying Machine Learning in Self-adaptive Systems: A Systematic Literature Review. *ACM Trans. Auton. Adapt. Syst.* **2021**, *15*, 1–37. https://doi.org/10.1145/3469440.
38. Lopez, P.A.; Behrisch, M.; Bieker-Walz, L.; Erdmann, J.; Flötteröd, Y.P.; Hilbrich, R.; Lücken, L.; Rummel, J.; Wagner, P.; Wiessner, E. Microscopic Traffic Simulation using SUMO. In Proceedings of the 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, 4–7 November 2018; pp. 2575–2582. https://doi.org/10.1109/ITSC.2018.8569938.
39. Almutairi, R.; Bergami, G.; Morgan, G. SimulatorBridgerDfT: A Real-Data Simulator for IoT-Osmotic Interactions. In Proceedings of the 2024 12th International Conference on Information Technology, Kuala Lumpur, Malaysia, 13–15 December 2024. https://doi.org/10.1145/3718391.3718429.
40. Gillgallon, R.; Bergami, G.; Almutairi, R.; Morgan, G. AI-Driven Multi-Agent Vehicular Planning for Battery Efficiency and QoS in 6G Smart Cities. *arXiv* **2025**, arXiv:2509.14877.
41. Dandekar, A.; Wegener, J.; Rahman, A.; Schulz-Zander, J. Towards Application Level Energy Monitoring for Green 6G Networks. In Proceedings of the 4th ACM Workshop on 5G and Beyond Network Measurements, Modeling, and Use Cases, 5G-MeMU '24, Los Angeles, CA, USA, 9–12 December 2024; Association for Computing Machinery: New York, NY, USA, 2024; pp. 8–13. https://doi.org/10.1145/3694810.3700158.
42. Konstantoulas, I.; Loi, I.; Tsimas, D.; Sgarbas, K.; Gkamas, A.; Bouras, C. A Framework for User Traffic Prediction and Resource Allocation in 5G Networks. *Appl. Sci.* **2025**, *15*, 7603. https://doi.org/10.3390/app15137603.
43. Tealab, M.; Hassebo, A.; Dabour, A.; AbdelAziz, M. Smart Cities Digital transformation and 5G–ICT Architecture. In Proceedings of the 2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA, 28–31 October 2020; pp. 0421–0425. https://doi.org/10.1109/UEMCON51285.2020.9298156.
44. Mahomed, A.S.; Saha, A.K. Unleashing the Potential of 5G for Smart Cities: A Focus on Real-Time Digital Twin Integration. *Smart Cities* **2025**, *8*, 70. https://doi.org/10.3390/smartcities8020070.
45. Byers, C.C. Architectural Imperatives for Fog Computing: Use Cases, Requirements, and Architectural Techniques for Fog-Enabled IoT Networks. *IEEE Commun. Mag.* **2017**, *55*, 14–20. https://doi.org/10.1109/MCOM.2017.1600885.
46. Schüler, C.; Gebauer, T.; Patchou, M.; Wietfeld, C. QoE Evaluation of Real-Time Remote Operation with Network Constraints in a System-of-Systems. In Proceedings of the 2022 IEEE International Systems Conference (SysCon), Montreal, QC, Canada, 25–28 April 2022; pp. 1–8. https://doi.org/10.1109/SysCon53536.2022.9773943.
47. Jörke, P.; Gebauer, T.; Böcker, S.; Wietfeld, C. Scaling Dense NB-IoT Networks to the Max: Performance Benefits of Early Data Transmission. In Proceedings of the 2022 IEEE 95th Vehicular Technology Conference: (VTC2022-Spring), Helsinki, Finland, 19–22 June 2022; pp. 1–7. https://doi.org/10.1109/VTC2022-Spring54318.2022.9860611.
48. Liu, X.; Ma, T.; Tang, Z.; Qin, X.; Zhou, H.; Shen, X.S. UltraStar: A Lightweight Simulator of Ultra-Dense LEO Satellite Constellation Networking for 6G. *IEEE/CAA J. Autom. Sin.* **2023**, *10*, 632–645. https://doi.org/10.1109/JAS.2023.123084.
49. Moussaoui, M.; Bertin, E.; Crespi, N. 5G shortcomings and Beyond-5G/6G requirements. In Proceedings of the 2022 1st International Conference on 6G Networking (6GNet), Paris, France, 6–8 July 2022; pp. 1–8. https://doi.org/10.1109/6GNet54646.2022.9830439.
50. Choi, T.; Kanno, I.; Ito, M.; Chen, W.Y.; Molisch, A.F. A Realistic Path Loss Model for Cell-Free Massive MIMO in Urban Environments. In Proceedings of the GLOBECOM 2022-2022 IEEE Global Communications Conference, Rio de Janeiro, Brazil, 4–8 December 2022; pp. 2468–2473. https://doi.org/10.1109/GLOBECOM48099.2022.10001398.
51. Kassam, J.; Castanheira, D.; Silva, A.; Dinis, R.; Gameiro, A. A Review on Cell-Free Massive MIMO Systems. *Electronics* **2023**, *12*, 1001. https://doi.org/10.3390/electronics12041001.
52. Xing, J.; Lv, T.; Li, W.; Ni, W.; Jamalipour, A. Joint Optimization of Beamforming and Noise Injection for Covert Downlink Transmissions in Cell-Free Internet of Things Networks. *IEEE Internet Things J.* **2024**, *11*, 10525–10536. https://doi.org/10.1109/JIOT.2023.3326275.
53. Mukherjee, S.; Lee, J. Edge Computing-Enabled Cell-Free Massive MIMO Systems. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 2884–2899. https://doi.org/10.1109/TWC.2020.2968897.

54. Ammar, H.A.; Adve, R.; Shahbazpanahi, S.; Boudreau, G.; Srinivas, K.V. User-Centric Cell-Free Massive MIMO Networks: A Survey of Opportunities, Challenges and Solutions. *IEEE Commun. Surv. Tutor.* **2022**, *24*, 611–652. https://doi.org/10.1109/COMST.2021.3135119.
55. Zhang, Y.; Di, B.; Zhang, H.; Lin, J.; Xu, C.; Zhang, D.; Li, Y.; Song, L. Beyond Cell-Free MIMO: Energy Efficient Reconfigurable Intelligent Surface Aided Cell-Free MIMO Communications. *IEEE Trans. Cogn. Commun. Netw.* **2021**, *7*, 412–426. https://doi.org/10.1109/TCCN.2021.3058683.
56. Voinov, I.A.; Chung, J.; Kettimuthu, R.; Bordel, B.; Alcarria, R.; Robles, T. Towards 6G Networks for Rural Environments: Vision for Improving Digital Inclusion through Open Source Hardware and Software. In Proceedings of the 2022 IEEE 40th Central America and Panama Convention (CONCAPAN), Panama City, Panama, 9–12 November 2022; pp. 1–6. https://doi.org/10.1109/CONCAPAN48024.2022.9997739.
57. Elgharbi, S.E.; Iturralde, M.; Dupuis, Y.; Gaugue, A. LoRaCAPS: Congestion-Aware Path Selection Protocol for Offshore LoRaWan Networking. In Proceedings of the 2024 20th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Paris, France, 21–23 October 2024; pp. 463–468. https://doi.org/10.1109/WiMob61911.2024.10770478.
58. Zhao, L.; Wang, C.; Zhao, K.; Tarchi, D.; Wan, S.; Kumar, N. INTERLINK: A Digital Twin-Assisted Storage Strategy for Satellite-Terrestrial Networks. *IEEE Trans. Aerosp. Electron. Syst.* **2022**, *58*, 3746–3759. https://doi.org/10.1109/TAES.2022.3169130.
59. Habeeb, F.; Alwasel, K.; Noor, A.; Jha, D.N.; AlQattan, D.; Li, Y.; Aujla, G.S.; Szydlo, T.; Ranjan, R. Dynamic Bandwidth Slicing for Time-Critical IoT Data Streams in the Edge-Cloud Continuum. *IEEE Trans. Ind. Inform.* **2022**, *18*, 8017–8026. https://doi.org/10.1109/TII.2022.3169971.
60. Said, O. A bandwidth control scheme for reducing the negative impact of bottlenecks in IoT environments: Simulation and performance evaluation. *Internet Things* **2023**, *21*, 100682. https://doi.org/10.1016/j.iot.2023.100682.
61. Orsini, G.; Posdorfer, W. Saving bandwidth and energy of mobile and IoT devices with link predictions. *J. Ambient. Intell. Humaniz. Comput.* **2021**, *12*, 8229–8240. https://doi.org/10.1007/s12652-020-02557-z.
62. Hagiwara, K.; Li, Y.; Sugaya, M. Weighted Load Balancing Method for Heterogeneous Clusters on Hybrid Clouds. In Proceedings of the 2023 IEEE International Conference on Edge Computing and Communications (EDGE), Chicago, IL, USA, 2–8 July 2023; pp. 171–176. https://doi.org/10.1109/EDGE60047.2023.00035.
63. Gillgallon, R.; Bergami, G.; Morgan, G. Testing Routing Strategies by Simulating the Mobile IoT Edge/Cloud Continuum. In Proceedings of the 2024 IEEE International Smart Cities Conference (ISC2), Pattaya, Thailand, 29 October–1 November 2024; pp. 1–6. https://doi.org/10.1109/ISC260477.2024.11004296.
64. Petrović, T.; Vidaković, A.; Doknić, I.; Veinović, M.; Bojović, Ž. An Adaptive Application-Aware Dynamic Load Balancing Framework for Open-Source SD-WAN. *Sensors* **2025**, *25*, 5516. https://doi.org/10.3390/s25175516.
65. Davies, A. IOT, Smart Technologies, Smart Policing: The Impact for Rural Communities. In *Smart Village Technology*; Springer: Cham, Switzerland, 2020; pp. 25–37. https://doi.org/10.1007/978-3-030-37794-6_2.
66. Ameddah, M.A.; Das, B.; Almhana, J. Cloud-Assisted Real-Time Road Condition Monitoring System for Vehicles. In Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 9–13 December 2018; pp. 1–6. https://doi.org/10.1109/GLOCOM.2018.8647334.
67. Orsini, G.; Bade, D.; Lamersdorf, W. CloudAware: Empowering context-aware self-adaptation for mobile applications. *Trans. Emerg. Telecommun. Technol.* **2018**, *29*, e3210. https://doi.org/10.1002/ett.3210.
68. Laurila, J.; Gatica-Perez, D.; Aad, I.; Blom, J.; Bornet, O.; Do, T.; Dousse, O.; Eberle, J.; Miettinen, M. From big smartphone data to worldwide research: The Mobile Data Challenge. *Pervasive Mob. Comput.* **2013**, *9*, 752–771. https://doi.org/10.1016/j.pmcj.2013.07.014.
69. Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, San Francisco, CA, USA, 13–17 August 2016; Association for Computing Machinery: New York, NY, USA, 2016; pp. 785–794. https://doi.org/10.1145/2939672.2939785.
70. Tibshirani, R. Regression shrinkage and selection via the lasso: a retrospective. *J. R. Stat. Soc. Ser. B (Stat. Methodol.)* **2011**, *73*, 273–282. https://doi.org/10.1111/j.1467-9868.2011.00771.x.
71. Dijkstra, E.W. A note on two problems in connexion with graphs. *Numer. Math.* **1959**, *1*, 269–271. https://doi.org/10.1007/BF01386390.
72. Almutairi, R.; Gillgallon, R.; Bergami, G.; Morgan, G. Approximating Real-Time IoT Interaction Through Connection Counting: A QoS Perspective. In Proceedings of the 2024 20th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Paris, France, 21–23 October 2024; pp. 260–265. https://doi.org/10.1109/WiMob61911.2024.10770322.
73. Du, X.; Chen, K.; Qiao, Z. A Multi-Objective Test Scenario Prioritization Method Based on UML Activity Diagram. *J. Electron. Test.* **2025**, *41*, 147–171. https://doi.org/10.1007/S10836-025-06173-7.
74. Bonica, R.; Pignataro, C.; Touch, J. A Widely Deployed Solution to the Generic Routing Encapsulation (GRE) Fragmentation Problem. *RFC* **2015**, *7588*, 1–12. https://doi.org/10.17487/RFC7588.
75. Russell, S.J.; Norvig, P. *Artificial Intelligence—A Modern Approach, Third International Edition*; Pearson Education: London, UK, 2010.

76. Xu, J.; Parnas, D. Scheduling processes with release times, deadlines, precedence and exclusion relations. *IEEE Trans. Softw. Eng.* **1990**, *16*, 360–369. https://doi.org/10.1109/32.48943.
77. Maurya, P.; Ramírez-Arroyo, A.; Sorensen, T.B. Cellular-Satellite Multi-Connectivity with Link Activation Based on Random Forest Classifier. In Proceedings of the 2024 20th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Paris, France, 21–23 October 2024; pp. 587–592. https://doi.org/10.1109/WiMob61911.2024.10770436.
78. Jiménez, E.; Gordillo, A.; Calero, C.; Ángeles Moraga, M.; García, F. Does the compiler or interpreter version influence the energy consumption of programming languages? *Sci. Comput. Program.* **2025**, *243*, 103270. https://doi.org/10.1016/j.scico.2025.103270.
79. Vallée, A. Digital Twins for Personalized Medicine Require Epidemiological Data and Mathematical Modeling: Viewpoint. *J. Med. Internet Res.* **2025**, *27*, e72411. https://doi.org/10.2196/72411.
80. Bergami, G. Towards automating microservices orchestration through data-driven evolutionary architectures. *Serv. Oriented Comput. Appl.* **2024**, *18*, 1–12. https://doi.org/10.1007/s11761-024-00387-x.