The Path Towards a Criticality-Sensitive Smart City

Rui Eduardo Lopes*†, Duarte Raposo†, Rodrigo Martins†, Diogo Mendes† and Susana Sargento*†
*Universidade de Aveiro, Aveiro, Portugal, {ruieduardo.fa.lopes, diogovmendes, susana}@ua.pt
†Instituto de Telecomunicações, Aveiro, Portugal, {dmgraposo, rodrigomartins}@av.it.pt

***Abstract*—The accelerated deployment of smart cities has led to a proliferation of diverse services operating simultaneously within urban environments. However, this growth presents the challenge of scalability issues, due to the varied resources used per service and the difficulty in analyzing wide-ranging network flows, since most traffic is not classified by their service. We present a path toward a smart city sensitive to criticality by: (1) showcasing contributions to evolve an infrastructure with the internet-of-things (IoT) and edge computing to accommodate traffic differentiation and service criticality; and (2) proposing a distributed network protocol, *Rank*, that is capable of dynamically allocating various types of resources along paths between nodes in the network infrastructure, so that new services could be added on demand. Our results show that it is possible to transform a smart city into a criticality-sensitive one by adopting a traffic classification with little to no impact on packet forwarding and time synchronization through hierarchical elements with respect to a GNSS source, where *Rank* conveniently solves the dynamics of variable traffic with low overhead.**

***Index Terms*—Mixed-Criticality, Resource Management, Edge Computing, Smart Cities, Time-Sensitive Networking**

I. Introduction

The global landscape is witnessing an accelerating deployment of smart cities. This rapid adoption has led to a proliferation of diverse services operating simultaneously within urban environments [1], that is also one of the central topics of debate for sustainable development in relatable communities and events such as the 2025 World Expo in Osaka, Japan[1] [2].

Not all smart cities have an infrastructure with internet-of-things (IoT) devices or edge computing. Among those that do, this increase of services and the constant rise in relevance of these infrastructures flag two problems that are not properly taken into account: (1) the increase in provided functions may not be scalable due to the variable set of types of resources used per service; and (2) the wide scope of ingress network flows makes the flows' analysis difficult, since most traffic is not commonly pre-classified.

In this paper, we showcase our approach to accommodate traffic differentiation and service criticality in a flexible and easily-adaptable manner. Then, we propose a distributed network protocol capable of discovering and dynamically allocating resources (in several types) along paths across the network infrastructure nodes. The results show that the application of automatic traffic classification for traffic differentiation, and time synchronization to keep a consistent data age throughout all systems and nodes pose little to no consequences to traffic forwarding. Our proposal for resource management ensures a linear time complexity as a worst-case scenario, and a low message overhead for establishing a resource allocation session. All these conclusions allow us to pave the way towards a criticality-sensitive smart city.

This paper continues in section II with a brief contextualization of the need for a future smart city deployment sensitive to service criticality, its requirements, and the path chosen for our proposed solution. With this assessment, in section III we reveal how the two main requirements are addressed: traffic classification and node synchronization. In section IV, we move forward to enable guarantees and dynamism in the use of node capabilities through the proposal of a fully distributed protocol for resource allocation based on admission requests, whose future integration in a Linux-oriented infrastructure is detailed in section V. We conclude with section VI, with several open challenges and future directions for research.

II. A Criticality-sensitive Smart City

The growth of any network infrastructure may evolve in the number of services that benefit from it. However, such growth cannot be permanent, as it will quickly reach limits due to the low diligence on the operation of resources allocated to the different requested activities. For this reason, as a main requirement to move forward with new solutions that allow better traffic management, it is important that an infrastructure gets to work optimally with existing traffic, capable to differentiate flows in the set of the various services and applications.

With traffic differentiation, we can determine the best conditions for transmitting each service. However, in reality, this quality-of-service (QoS) still lacks a requirement that is rarely considered in networks, which is *time*. As cities evolve toward machine-to-machine (M2M) communication scenarios and vehicular autonomy, smart cities are entering an evolutionary path similar to that of industry, moving rapidly toward large-scale automation across various city services. Time, in these communications, can be the highest priority variable among all others, meaning that traffic designation must first pass through its characterization as synchronous or asynchronous traffic.

Therefore, a mechanism or set of techniques are required to create an environment where time is the primary consideration in terms of network traffic classification in an network infrastructure. Coming from industry, there are already several solutions that have been adopted over time such as time-triggered Ethernet, PROFINET, or EtherCAT, but all of these fall short because they are vendor-specific or closed for industrial solutions [3]. Given that, in smart city environments we have general-purpose traffic that may require proper real-time

[1]Future City Pavilion (Accessed in August 1, 2025) https://www.expo2025.or.jp/en/future-index/future-life/city/.

978-3-903176-75-1

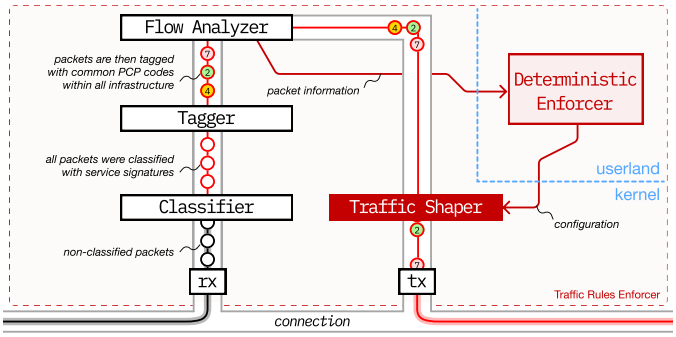

Fig. 1. Network traffic classification at the entry point of a smart city network access node.

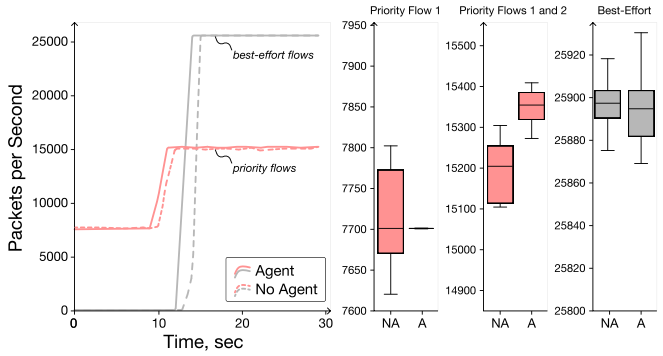

Fig. 2. Results of two concurrent priority flows experiment. For both priority and best-effort traffic, *A* stands for with agent, and *NA* for without agent.

considerations, we adopted time-sensitive networks (TSNs) as our base solution.

TSN is not a network protocol, but rather a set of network mechanisms and protocols that are applicable as an extension to Ethernet [4]. Being regulated by IEEE, TSN capabilities allow greater flexibility than other industrial alternatives, in an open manner, both in communication between various machines, in their dynamic configuration, as well as in the low cost of adoption as it is an extension to Ethernet implementation, which is already a common network access technology in smart city infrastructures [3], [4].

III. Classification and Synchronization

Time is therefore a primary variable in our process of making a smart city sensitive to the criticality of its services. In order to accommodate a TSN solution in the context of a smart city, we first need to establish some rules over the traffic that an infrastructure already has, or over others that may be added throughout time.

We need to proceed with a first phase in which we *classify* the traffic, so that we can parameterize, in the future, all characteristics per service according to our deployment. Serving to provide an output of a set of service descriptors, this classification must be executed in the network infrastructure access layer nodes. Figure 1 depicts our agent solution for the classification, and its application to ingress traffic in the smart city context.

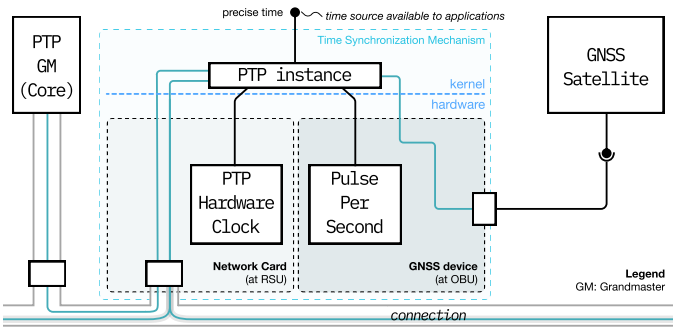

Fig. 3. Time synchronization with PTP grandmaster via RSUs and GNSS with PPS via OBUs.

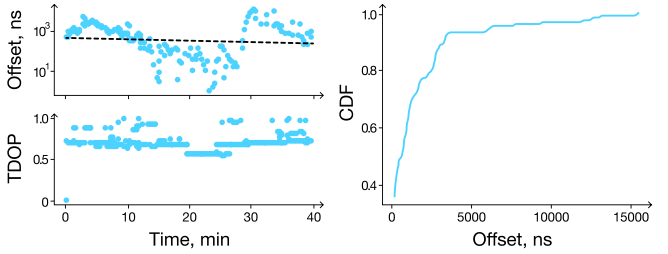

Fig. 4. Results on time synchronization with PTP and GNSS, with PPS, on a mobility scenario.

The solution in figure 1 shows a classification procedure that, in a first phase, identifies the ingress traffic of a node in the access layer of the infrastructure, tagging packets with a priority code point (PCP) value in a range that is common to all nodes of the infrastructure. Once the packets have been tagged with the appropriate priority in the context of the network they entered, a description of the component flows is updated. This will serve as input for a component that will determine if there are configurations to apply, so that the traffic is subject to a shaping technique.

Figure 2 shows the results of two concurrent priority flows with a best-effort one in which it is revealed the effect of our classification agent has in turning priority flows more stable, leaving best-effort to become more variable, as expected.

The traffic shaping that can be subject to the procedure in figure 1 may affect bandwidth or time, in the latter case configuring its own scheduler so that it adjusts in relation to an open transmission time. For this configuration to be applied, a second requirement, of synchronization, must be applied to all nodes involved in this infrastructure.

In a previous work of ours in [5], an integrated hierarchy for a city was proposed to enable full synchronization of the various network elements of a smart city with the highest possible clock precision. Through a combination of precision time protocol (PTP) and global navigation satellite system (GNSS), it was possible to study an approach that would maintain synchronization between various partners based on a reference clock. We also evaluated the possibility of synchronization tools such as pulse per second (PPS) in GNSS, as we can see in figure 3.

Figure 4 showcases the results of [5] on the application of GNSS with PPS as a main clock source coming from a roadside unit (RSU) or via a GNSS antenna at an on-board unit (OBU). From the results, it is possible to conclude that, having the time dilution of precision (TDOP) in ideal conditions during the entirety of the experiments (TDOP is ideal if below or equal to 1.0), the clock offset got closer to 10^2 ns. These results were obtained in a mobility scenario with both environments of city and highway, with speeds varying from a max of 50 km/h to a max of 120 km/h, in the 15 to 20 and 25 to 30 minute ranges. With these conclusions we establish an infrastructure knowledgeable of its traffic patterns and can proceed to the integration of new configurations.

IV. Ease and Requirements of Configuration

With classification and synchronization properly designed and implemented in a smart city infrastructure, the path opens up to enable the integration of new services [6]. With services properly described in their various characteristics in terms of traffic, time, and even computational resources, we shall be able to solve the problem of dynamically allowing the inclusion of new rules in paths between any two points in the network, so that the infrastructure can accommodate, in a guaranteed way, a new service or function.

There is no reference topology for a city's network infrastructure, but this composition shall be viewed as a set of heterogeneous nodes. This increases the difficulty in researching a solution, given that not only does the resource allocation problem already exist, but the heterogeneity requires that, beforehand, the availability of each involved node must be estimated according to an admission request that is established in a network session.

To solve these problems, we propose *Rank* as a fully-distributed network protocol that enables the discovery, estimation, allocation, and sharing of resources based on admission requests from a requester to a destination (listener). The paths between the two points cannot be designated by the client, but rather discovered: a task which is performed based on the availability of each node as the admission request passes between pairs of nodes toward the intended listener.

In the current state-of-the-art, resource management protocols featuring resource allocation and estimation functionalities lack three main capabilities in simultaneous [7], [8], [9], [10]: i) the ability to extend the list of resource types considered for assessment and allocation; ii) the ability to modify the resource assessment function by computational node; and iii) the ability to describe such a new resource type in a dynamic but still standard schema. Regarding the latter, *Rank* only acknowledges requirements described via standardized YANG modules, that are universally adopted descriptions of resource specifications [11].

To be fully functional, *Rank* is comprised of a set of six messages, as seen in action in figure 5: the exclusive admission requests (EARs) and multiple admission requests (MARs) are the only requirement list carriers, as well as

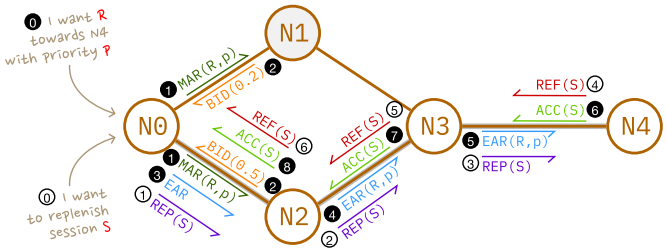

Fig. 5. Admission request (labeled by filled circles from 0 to 8) and replenishing with Rank (labeled by empty circles from 0 to 6).

of the priority of the session, being the trigger for the self-assessment. The BID message is a response to a MAR message in which a node shares its suitability value to the requesting node. The acceptation (ACC) and refusal (REF) messages are two possible responses to the Rank session. Finally, the replenishing (REP) message is the one responsible for the restoration of the resource allocation.

At each node visited by a *Rank* admission request, a resource capacity assessment is performed against the list of requested requirements [12]. From this evaluation, a suitability value, B, always emerges, which is a normalized value between 0 and 1, where 0 means total inability to fulfill the request, and any other value means being able to guarantee the request. When a node has more than one neighbor that provides a path to the intended destination, each of these should share its suitability value, and the highest of the set should identify the next node with which to continue the resource allocation session.

The assessment is structured around five core criteria, c_x. First, the node verifies whether its bare-metal hardware, c_1, is fundamentally capable of meeting the requested resources. If not, the node immediately disqualifies itself, granting a null suitability ($B = 0$). If the bare-metal check passes, then the node evaluates the availability of its current resources, c_2, weighting each requirement according to its order of importance within the request. If one resource fails to achieve availability, then, the node immediately disqualifies itself. In addition to resource-based evaluations, *Rank* incorporates a priority, c_3, for the request, assigning greater values to tasks with higher urgency. Network proximity, c_4, is also considered: the node estimates its connectivity towards the request's intended destination (hereby called listener), using network metrics such as latency or packet loss to judge how quickly and reliably the service can be delivered, even in the case that no network requirements were described in the admission request. Lastly, *Rank* accounts for historical performance, c_5, rewarding nodes that have reliably handled similar tasks in the past. The final suitability value is calculated by $B = c_1 \times c_2 \times c_3 \times \frac{c_4 \times c_5}{2}$, which is an expression that multiplies c_1, c_2, and c_3, and averages c_4 and c_5 scores due to their own subjective nature.

Preliminary tests have been conducted in a C++ custom-made emulated environment, where we could fine-tune conditions, that allowed us to draw several conclusions regarding

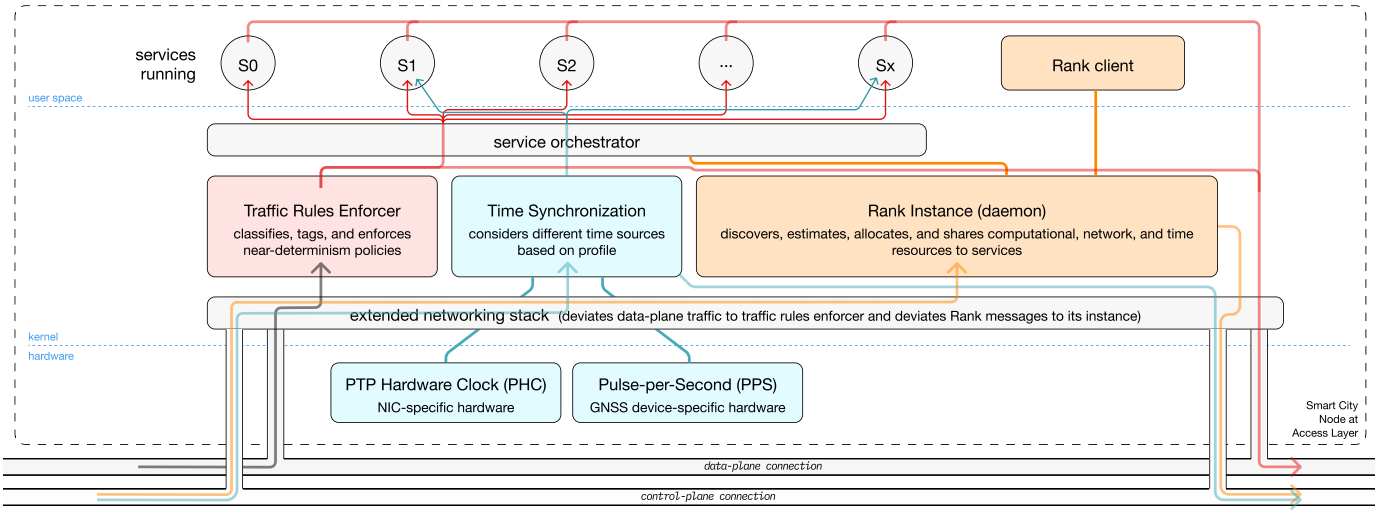

Fig. 6. Full integration of contributions in a smart city access layer node.

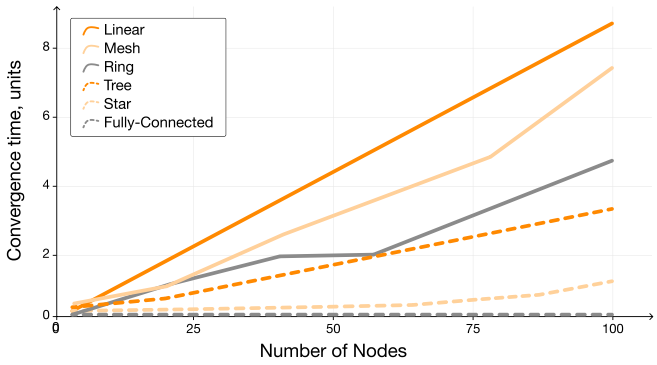

Fig. 7. Rank converging time per topology and per number of nodes.

scalability criteria and how complexity is distributed over time across different network topologies. Linear, mesh, tree, ring, star, and fully-connected topologies were tested with connections of 100 Mbps each, and with a varying number of nodes ranging from a minimum acceptable per topology up to 100 nodes.

Focusing on how messages are sent between nodes, we observe that the worst case of temporal complexity occurs in the linear topology, as we can see in figure 7. Since *Rank* has control mechanisms for loops (supported by session identifiers used in the messages), it is very positive to obtain these results since message propagation occurs within the lowest possible complexity given that, unequivocally, a session will always have to pass through a minimum set of node pairs that establishes the shortest path for resource allocation.

Looking at the overhead caused by the *Rank* protocol messages, it is estimated that they have a mean overhead of 1.16 ± 0.13 kbit/s, in the selected nodes throughout path discovery. Compared to other similar protocols such as resource reservation protocol for traffic engineering (RSVP-TE) [13], the overhead of *Rank* messages is much lower, as it is restricted only to connections that affect nodes with links between a talker and the intended listener. This scenario, through admission requests, becomes more comparable to other similar protocols such as the TSN's resource allocation protocol (RAP) [14]. Nevertheless, the execution of *Rank*, being exclusive to itself, has a reduced footprint, since RAP requires the parallel use of other protocols such as the link-layer registration protocol (LRP) [14], with which it confirms responses to admission requests against fulfilled databases on the capabilities and state of each machine. *Rank* works more selectively, requiring only the needed information for the system receiving a message, and thus preserving a guaranteed pre-reservation until confirmation or rejection from the listener.

Rank also includes heartbeat messages that will allow it to maintain awareness of the activity of the established reservations, node-to-node, using a minimum message size (only the header length of a *Rank* message, with 17 bytes). The overhead of this type of operation is lower than the previously estimated impact, given that the heartbeat message, although periodic, has a maximum length of about one-third of the most common message evaluated for admission requests.

V. Integrated Approach

It is possible to transform a smart city infrastructure with IoT devices and edge computing into a criticality-sensitive smart city, as in figure 6. Taking as an essential starting point the adoption of criteria that allow for the classification of existing traffic to enable service differentiation, and the use of well-adapted mechanisms for temporal synchronization of all network elements, the distance to our objective becomes very short, having only to deal with the natural dynamics of the variable traffic of an infrastructure of this type.

Unlike our main reference of industrial networks, smart cities are strongly characterized by having a very inconsistent network profile. Knowing how to handle the entry of new services in operation (as well as the breakdown of others) with

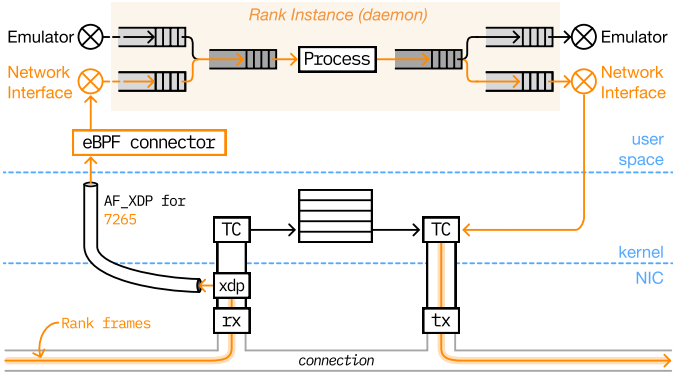

Fig. 8. Integration of Rank within Linux operative system

network configurations already in place (especially assuming temporal shaping configurations, such as time-aware shaping (TAS) scheduling in TSN) is a task that our *Rank* proposal conveniently solves with a low overhead for its operation of approximately 1.2 kbit/s in a very punctual manner, oriented to an admission request, and selecting only nodes that are capable of providing access to the service recipient, which is a better performance in relation to common standard solutions.

Current efforts are being made in integrating *Rank* as low as possible in the Linux operative system[2][12]. *Rank* is designed to be working in the OSI Layer II, which requires frames to be properly identified with a specific EtherType value. Coded as `7265`, *Rank* will natively fail to be recognized by the operative system as soon as it starts parsing procedures in the networking stack. To solve this, and with the help of extended Berkeley packet filter (eBPF) directives, we are able to directly access frames in the network interface card (NIC), forwarding their content directly onto the *Rank* daemon that holds the responsibility to maintain a connection to a `AF_XDP` socket to allow newly received data to come. Figure 8 depicts this integration of *Rank* as a standalone solution to be installed in each surrogate node that is aimed to have its computing, network, and time resources widely at disposal of city services.

VI. Conclusions and Future Work

Our proposed approach showcases a first big step into solutions that can finally give a sense of criticality and time within an urban scenario, as a close-future hybrid environment of autonomous vehicles, interconnected things, and legacy entities. Besides our current contributions, there is still a large gap in the availability of open-source alternatives for TSN protocol and its mechanisms so that, even with *Rank*, common operative systems such as Linux are able to actuate over the system and deploy the requested resources for TSN-based requirements.

In a larger scope of smart cities infrastructures, there are still reliability concerns through the application of redundant paths or guarantees of content delivery between network nodes. This goal is being achieved by the application of TSN criteria and considerations in wireless environments, but slowly since as regulatory bodies for wireless network access technologies are different, there is a low coordination between common efforts.

With some efforts from the scientific community, services within a smart city, if virtualized, can be instantiated just-in-time and with a shared time domain, as well with strict computing, network, and timing requirements in place. For this reason, the application of TSN schedules and configurations under orchestration and virtualization environments is still an open challenge that brings a closing future direction for research.

[2]Project source code in https://doi.org/10.5281/zenodo.15482200.

Acknowledgments

This work is supported by the European Union/Next Generation EU, through Programa de Recuperação e Resiliência (PRR) [Project Nr. 11: New Space Portugal (02/C05-i01.01/2022.PC644936537-00000046)], by the European Union Programa FEDER, Operação n.º 14795 - COMPETE2030-FEDER-00929900, and Fundação para a Ciência e Tecnologia within grant FCT.2021.06223.BD.

References

[1] B. Cengiz, I. Y. Adam, M. Ozdem, and R. Das, "A survey on data fusion approaches in iot-based smart cities: Smart applications, taxonomies, challenges, and future research directions," *Information Fusion*, vol. 121, p. 103102, 2025.

[2] Bureau International des Expositions, *Expo 2025 Osaka, Kansai, Japan, Official Guidebook*. JTB Publishing, 978-4533165054, 2025.

[3] Y. Seol, D. Hyeon, J. Min, M. Kim, and J. Paek, "Timely survey of time-sensitive networking: Past and future directions," *IEEE Access*, vol. 9, pp. 142506–142527, 2021.

[4] "Ieee standard for local and metropolitan area networks–bridges and bridged networks," *IEEE Std 802.1Q-2022 (Revision of IEEE Std 802.1Q-2018)*, pp. 1–2163, 2022.

[5] R. Martins, D. Raposo, R. Lopes, P. Rito, and S. Sargento, "Time synchronization in v2x communications," in *2024 IEEE Vehicular Networking Conference (VNC)*, pp. 25–32, 2024.

[6] J. Pires, V. K. Shukla, L. Wanganoo, and S. Vyas, *Challenges and Opportunities in Smart City Network Management Through Blockchain and Cloud Computing*, ch. 7, pp. 111–127. John Wiley & Sons, Ltd, 2023.

[7] C. Rublein, F. Mehmeti, T. D. Gunes, S. Stein, and T. F. La Porta, "Scalable resource allocation techniques for edge computing systems," in *2022 International Conference on Computer Communications and Networks (ICCCN)*, pp. 1–10, 2022.

[8] M. M. I. Khan and G. Nencioni, "Resource allocation in networking and computing systems: A security and dependability perspective," *IEEE Access*, vol. 11, pp. 89433–89454, 2023.

[9] J. Bachiega, B. Costa, L. R. Carvalho, M. J. F. Rosa, and A. Araujo, "Computational resource allocation in fog computing: A comprehensive survey," *ACM Comput. Surv.*, vol. 55, jul 2023.

[10] A. Mahapatra, K. Mishra, R. Pradhan, and S. K. Majhi, "Next generation task offloading techniques in evolving computing paradigms: Comparative analysis, current challenges, and future research perspectives," *Archives of Computational Methods in Engineering*, vol. 31, pp. 1405–1474, Apr 2024.

[11] M. Björklund, "The YANG 1.1 Data Modeling Language." RFC 7950, Aug. 2016.

[12] R. E. Lopes, D. Raposo, P. V. Teixeira, and S. Sargento, "Self-assessment approach for resource management protocols in heterogeneous computational systems," *arXiv Distributed, Parallel, and Cluster Computing (cs.DC)*, vol. 2508.02202, 2025.

[13] D. O. Awduche, L. Berger, D.-H. Gan, T. Li, D. V. Srinivasan, and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels." RFC 3209, Dec. 2001.

[14] IEEE, "P802.1qdd – resource allocation protocol." https://1.ieee802.org/tsn/802-1qdd/, June 2024. (Accessed on 08/02/2025).