Time-Sensitive Networking on virtualized network components

Simon Burger, Florian Wiedner*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: simon.burger@tum.de, wiedner@net.in.tum.de

***Abstract*—Time-Sensitive Networking (TSN) and network virtualization are integral to our modern networking landscape. While TSN enhances Ethernet with mechanisms for deterministic data transmission, network virtualization provides the flexibility and scalability required for rapidly deploying applications. However, their convergence remains a challenging topic.**

This paper surveys the individual challenges of TSN and network virtualization before deriving issues that complicate the integration of TSN in virtualized networks (e.g., timing, scheduling, and hardware constraints). It also gives an overview of possible synergetic effects of TSN in virtualized environments before concluding with potential research directions that would benetif effective integration.

***Index Terms*—time-sensitive networking, tsn, network virtualization, tsn in virtualized networks**

1. Introduction

Time-Sensitive Networking (TSN) can be seen as the critical enabler of real-time networking. Industrial and automotive systems, as well as communication networks, have increasingly become real-time applications [1]. With this, the real-world need for deterministic communication with strict latency and jitter guarantees has increased drastically. For example, a missed deadline in an airbag control system is a serious safety hazard.

TSN extends the Ethernet standard to support such guarantees and aims to enable the convergence of time-critical and best-effort network traffic [2]. Simultaneously, network virtualization has become irreplaceable. It enables flexible and programmable network infrastructures that allow virtual networks to share the same physical components [3]. This allows for separating physical infrastructure and network services, which is essential for technologies like 5G, edge computing, and for modern data center environments, as agile deployment and multitenancy are indispensable [4], [5]. Although these technologies developed independently, their combination promises appealing opportunities. However, successfully integrating them is a challenging task. While virtualized environments provide flexibility, scalability, and resource efficiency, they can undermine the timing determinism that TSN was designed to guarantee [3], [6]. We will investigate the challenges in deploying TSN within virtualized network environments. Furthermore, we will discuss what is currently missing to make this process seamless. The paper is structured as follows: Section 2 provides the necessary background on TSN and network virtualization concepts. Section 3 focuses on challenges intrinsic to TSN, whereas Section 4 then outlines the challenges of virtualized networks. Lastly, in Section 5, we explore the integration challenges, highlighting conflicts and mutual benefits that arise when attempting to guarantee deterministic communication over a virtualized infrastructure, before concluding by summarizing the findings and suggesting directions for future research in Section 6.

2. Background

To grasp how TSN may converge with virtual networks in the future, it is necessary to understand each technology separately first. Therefore, this section briefly outlines essential mechanisms of TSN and network virtualization.

2.1. Time-Sensitive Networking

TSN is a set of standards concerning deterministic communication that builds upon Ethernet, as Ethernet can not guarantee latency or low jitter (i.e., variation in packet delay) [1]. Essential concepts include:

- *Scheduling* means to decide which frames to transmit at what time. While numerous optimization objectives exist, latency and determinism are the most relevant ones for TSN [1].
- *Traffic shaping* is done by prioritizing frames and delaying the queuing of others. Combined with a fitting schedule, this establishes a traffic profile that complies with Quality of Service (QoS) requirements (e.g., latency) [1].
- The *Precision Time Protocol* (PTP) is used to synchronize clocks of each node in a system by choosing a Grandmaster Clock, which propagates timing information to all other clocks [7].

2.2. Virtualization

Network virtualization is used to create multiple virtual networks (VNs) on shared physical infrastructure, which is referred to as the substrate network [3]. The mapping of virtual nodes and links onto the physical substrate is known as Virtual Network Embedding (VNE) [8].

Different virtualization techniques are used to create virtualized environments. These include virtual machines (VMs), bare-metal hypervisors, and containers, each with different performance and flexibility properties [4]. In a virtual network, these can constitute virtual nodes. Regular

 doi: 10.2313/NET-2025-11-3_13

Switches, Network Interface Cards (NICs), and Routers can be virtualized and part of a VN. As these nodes are independent of their physical location, they can be placed anywhere, which impacts resource usage, load balancing, and other aspects [9]. By leveraging kernel bypassing techniques (e.g., Data Plane Development Kit (DPDK)), the virtualization overhead caused by the kernel networking stack can be reduced. Here, network device drivers implemented in the userspace are used to send or receive data directly, avoiding the kernel networking stack. [10]. Furthermore, with direct device assignment, VMs can avoid any hypervisor involvement and access I/O network devices directly, thus reducing virtualization overhead [11].

3. Challenges in Time-Sensitive Networking

Modeling and simulating time-sensitive networks. Network modeling is essential for developing and testing novel frameworks, architectures, or protocols. For real-time networks in particular, a robust validation is inevitable. A tight analysis is fundamental to achieving given requirements with minimal resources (i.e., better utilization). For instance, several different modeling languages already exist for automotive embedded systems capable of modeling high bandwidth Ethernet connections. However, Ashjaei et al. note that most of the existing open-source solutions (e.g., AMALTHEA4public, COMDES) cannot support the requirements of TSN [12].

Performance simulation, which can be done in hardware (HW) or software, is another crucial but difficult topic. Due to only a few studies in this field, existing TSN performance simulation methods, especially physical simulation, while still beneficial, are not yet fully mature and must be improved further (e.g., timing precision) [13]. Their ideal usecase can be seen in validating network characteristics under various conditions and enabling rapid testing of different TSN parameters (e.g., within TSN schedulers).

Clock synchronization. To enable real-time communication in a convergent network, both end devices and network devices must have the same notion of time [14].

Firstly, the PTP relies on precise time stamps of each message. Less precision in clock synchronization induces jitter and decreases determinism [15].

Secondly, once a common understanding of time has been established, time is crucial for assigning transmission time slices to packets of various priorities [2]. As discussed by Chahed et al., even minor misalignments can have a devastating effect on latency and jitter, possibly violating the given QoS requirements [1].

Reliability. Any deterministic system has to be reliable. In TSN, reliability is usually guaranteed by a combination of mechanisms. These include Path Control and Reservation, Frame Replication and Elimination for Reliability, and Per-Stream Filtering and Policing. These manage and reserve network resources along the chosen path, send replicated frames along disjoint paths, and filter traffic to prevent overload, improving network reliability. While necessary, these practices lead to increased complexity and resource overhead (e.g., bandwidth). Furthermore, it introduces a high risk of misconfiguration [13].

Scheduling and traffic shaping. In state-of-the-art network architectures, the delays for propagation, processing, and transmission of messages can be considered deterministic and constant. Therefore, the non-determinism is primarily a result of scheduling and queuing delays [14]. To minimize the jitter that a convergent real-time network experiences, appropriate scheduling strategies are essential. The schedule sets the ideal traffic profile (e.g., type and volume of network traffic), while the selected shaping mechanism ensures that the traffic adheres to said profile. Different shaping mechanisms have been developed, each with their respective use cases. Some mechanisms, like Asynchronous Traffic Shaping, do not require a centralized clock but may not be able to meet certain QoS requirements regarding latency. More common approaches like Cyclic Queuing and Forwarding and Time-Aware Shaping (TAS) require time synchronization across all network parts, which is, as previously described, challenging to achieve. Depending on the application area, you must weigh different goals (e.g., performance v. dynamism) to choose the best-fitting traffic-shaping and scheduling algorithms [1]. The transmission schedule in TSN networks is usually computed offline (i.e., all requests known a priori). While there are a range of proposed mechanisms (e.g., TAS), the problem can still not be solved in polynomial time, even if the arrival time of every time-triggered traffic is known in advance [6]. For this reason, heuristic solutions are often applied (e.g., HLS [16]). They do not deliver optimal schedules, but are faster and more efficient in complex networks and under restricted resources [17].

Hardware support. Specialized hardware can be expensive. Together with an increasing need for time-critical enabled networks, this has led to increased efforts towards achieving real-time capabilities using commercial off-the-shelf (COTS) hardware [7], [18] or software-based implementations [15], [19], [20]. Without specialized hardware, some precision will be lost. For example, if a NIC does not support the IEEE 802.1AS standard, the oftentimes required timing precision in the nanosecond range can not be achieved [7].

Moreover, clock synchronization is a significant difficulty for software-only implementations of the PTP. This is due to time stamps of messages being introduced in higher network layers instead of specialized hardware interfaces, leading to more jitter [15]. Especially in embedded systems, it can be necessary to implement specific time-sensitive functions directly in hardware to reduce load on the Central Processing Unit (CPU) [12].

On the other hand, some feasible and precise enough software-only implementations of the PTP do already exist [15]. Furthermore, newer Linux kernel versions are equipped with a TAS implemented in the TAPRIO (Time-aware priority shaper) queuing discipline (qdisc) [7]. Software-only frameworks for emulation and performance testing of TSN networks are also starting to be used in areas like mobile networks [13].

4. Challenges in Network Virtualization

Mapping of physical resources. The mapping of physical resources (i.e., the VNE problem) constitutes one of the most critical and complex challenges in network virtualization. This assignment of VN components

 doi: 10.2313/NET-2025-11-3_13

to the physical substrate must satisfy various constraints, such as CPU capacity, link bandwidth, and other QoS guarantees [21]. The VNE problem is shown to be NP-hard, both in offline settings and in more realistic online scenarios (i.e., requests arrive dynamically over time). For this reason, heuristic or approximation-based algorithms are employed to achieve feasible mappings. However, these often operate under simplifying assumptions such as infinite substrate capacities or static topologies, which limit their applicability in real-world settings [8].

Another dimension to the resource mapping challenge is the dynamic nature of virtualized environments. Virtual machines and containers may be migrated or scaled in response to changing requirements or workloads [22]. The virtual machine placement can be optimized for different objectives, like energy consumption or response time [9]. These changes require a new mapping of resources and impact the previously computed transmission schedules.

Moreover, virtualization technologies themselves introduce overhead that must be factored into the resource mapping in case of strict QoS requirements. In particular, hypervisor-based virtualization introduces additional latency and jitter. To mitigate these issues, recent approaches advocate kernel-bypassing techniques (e.g., DPDK) and direct device assignment [11]. Tail latency and throughput improvements can be an order of magnitude when utilizing these. [23].

Interoperability between heterogeneous networks. As virtual networks scale across administrative domains and physical infrastructures, interoperability issues between heterogeneous networks emerge.

The deficiency in standardized interfaces between infrastructure providers (InPs), service providers (SPs, i.e., organizations that create VNs on the infrastructure of different InPs), and end users is one issue [8]. Proprietary network orchestration implementations hinder cross-domain compatibility and complicate VN provisioning for the SPs [24]. A standardized model (e.g., XML-based) is required to express requirements, making processes like signaling and bootstrapping across diverse infrastructure domains seamless and simplifying the establishment of end-to-end virtual links [8].

Another issue is the naming and addressing incompatibility across network domains. Each virtualized environment may implement its own naming and addressing schemes. This heterogeneity further complicates end-to-end communication. Chowdhury et al., therefore, consider it an important research challenge to find a suitable framework for enabling global connectivity. They further mention one such framework called iMark, which would theoretically be viable but is not feasible in practice [8].

Resource discovery also becomes non-trivial, as inter-domain virtual link provisioning requires infrastructure providers to maintain and expose accurate representations of their physical topology and link capacities. This can be done event-based or periodically [8]. Cross-provider resource allocation mechanisms must be established to allow end-to-end virtual links spanning multiple administrative domains. Without it, VNs would be restricted to a single domain, which may not always be sufficient. This again necessitates standardized interfaces and interoperable signaling protocols between service and infrastructure providers. The diversity of physical networking technologies, each with different characteristics, makes this task particularly challenging [3], [8].

Security in VNs. Isolation is one of the design criteria network virtualization should fulfill. While VNs leverage mechanisms like secure tunnels and advanced encryption to achieve some degree of security, some threats on the physical layer still have to be addressed [3]. Examples would be different forms of physical tampering.

An important aspect in that regard is the monitoring and efficient isolation of failures within the substrate network [3]. Network Virtualization is equipped with several monitoring tools (e.g., for bandwidth or memory usage) that can be used to detect malfunctioning or malicious nodes, which have to be isolated and then removed from the network [25]. Moreover, the flexibility and programmability of each virtualized network element [3], [25] imply an increase in the importance of secure programming.

Lastly, to help prevent failures that could lead to compromised network security, the InPs are required to employ admission control. Primarily due to dynamic resizing of allocated resources, it is challenging to constantly account for the physical network's available resources. However, this is necessary for upholding QoS guarantees and ensuring a secure network [8].

5. Combining Time-Sensitive Networking and Network Virtualization

As these technologies mature, integrating one into the other is a logical step. This section will explore difficulties arising from said integration, like timing precision and scheduling, while highlighting synergetic effects, making this a promising area of research.

5.1. Challenges and limitations

Timing Precision and Hardware Constraints. In a TSN-enabled network, precise time synchronization across all network components is typically achieved by the PTP. However, in software-only implementations of the PTP, time stamps are obtained at higher network layers due to the lack of specialized hardware interfaces, introducing jitter and undermining timing precision [15]. This issue is particularly critical in virtualized environments because the often-used general-purpose hardware limits support for specialized functionality.

Virtualized environments add complexity due to the lack of hardware timestamping support and delays introduced by hypervisors and virtual switches [10], [15]. In such settings, achieving nanosecond precision is difficult. For example, using the software-based TAPRIO qdisc in Linux systems may result in synchronization failures when combined with the PTP if the NIC lacks IEEE 802.1AS support [7]. Frame drops or delays and clock synchronization become unpredictable when hardware queues are full, or timing mechanisms are unavailable or not properly emulated [6]. This additional unpredictability caused by virtualization can not be tolerated in real-time environments.

Moreover, VMs and containers introduce additional overhead through multiple network stack traversals, which

 doi: 10.2313/NET-2025-11-3_13

can considerably impact timing guarantees [10]. For example, it takes roughly 3 to 10 µs to enter and exit a VM and perform an I/O operation [26]. While the use of kernel-bypassing technologies like the DPDK and direct device assignment improve performance, they reduce flexibility by tightly coupling virtual machines to specific physical resources (i.e., complicating VM migration). They also come at the cost of increased CPU load, which can further jeopardize the strict QoS requirements of TSN [11].

Additionally, software-only TSN stacks are inefficient with regard to checksum calculations and memory operations, as specialized HW like NICs and Direct Memory Access controllers typically handle these. In such cases, these tasks consume considerable CPU resources and can introduce latency, which is detrimental to real-time performance and reliability. The software-based network stack with TSN support proposed by Denzler et al. tolerates the mentioned loss in performance in favor of reducing dependence on hardware-specific capabilities [27].

Lastly, heterogeneous networks and inconsistent hardware capabilities across network segments further worsen the synchronization issues. The need for priority translation and the degradation of synchronization across domains can result in inconsistent scheduling behavior and increased jitter, which are difficult to compensate for [1]. Depending on the given TSN requirements, these conditions may not be sufficient.

Scheduling Complexity and Resource Contention. TSN relies on centralized scheduling and shaping mechanisms (e.g., TAS). The schedules are usually computed offline, which requires a static view of the network topology and traffic flows [6]. These mechanisms become more complex when integrated into virtualized environments, where topologies are dynamic due to VM migrations or scaling. In such cases, offline scheduling approaches can become ineffective, as real-time dynamic reconfiguration is not scalable [6], [28].

Virtualization techniques amplify scheduling complexity as they allow virtual components such as talkers and listeners to be placed flexibly across the network nodes. This increases the dimensionality of the scheduling problem since both placement and timing must be optimized to preserve end-to-end latency guarantees [1].

Moreover, many incremental scheduling algorithms are designed to accommodate new flows without modifying previously allocated time slots to prevent jitter and inconsistencies. While these algorithms support some dynamic updates, it becomes problematic in virtualized networks where frequent and flexible reconfiguration is often required [14].

Additionally, optimal resource scheduling is itself, as discussed in section 3, NP-hard and typically requires heuristic solutions [8]. These heuristics may not be accurate enough to support the hard real-time requirements of TSN, even more so in scenarios where multiple infrastructure providers are required to coordinate themselves.

Highly parallel processing platforms, like in automotive systems, introduce even more uncertainty because multiple virtualized components compete for shared resources (e.g., memory, I/O bandwidth). This may lead to execution time variability (i.e., jitter), presenting another challenge in virtualized TSN environments [12]. The poor timing predictability is especially problematic for traffic with hard deadlines, meaning such virtualized parallel systems are not well-suited for real-time guarantees.

Finally, complexity further increases because of the physical resource constraints (e.g., limited processing or buffer capacity at gateways) to which deployed virtual network functions must adhere. It is not possible to minimize the resulting jitter in polynomial time [29].

5.2. Possible synergetic effects

Although they do not solve any of the challenges of implementing one another, TSN and network virtualization each address some weaknesses of the other. As previously mentioned, real-time-enabled virtualized networks have applications across a wide range of industries. The following will be a concise overview of how both technologies can benefit from each other once the previously discussed challenges are overcome.

- Virtualized networks may introduce unpredictable latency, jitter, and even packet loss due to shared physical resources and virtualization overhead [6], [11]. The lack of real-time guarantees can make VNs impracticable in ultra-low latency systems. If TSN is introduced to such systems, a bounded latency and deterministic transmission can be guaranteed through the various mechanisms described in the TSN standard.
- TSN is typically rather strict, assuming static topologies and centralized offline scheduling [6]. This can become infeasible in more dynamic environments, like cloud-based or industrial systems. Network virtualization can enable dynamic deployment of TSN-enabled functions across VMs, increasing flexibility and scalability.
- As TSN traditionally requires hardware support (e.g., by NICs [7]), experimenting with it can be expensive. Many TSN features can be implemented in software on general-purpose COTS HW, enabling better academic research and simpler testing using virtualized networks.

6. Conclusion and future work

The paper examined TSN and network virtualization separately before investigating the challenges of deploying TSN within virtualized network environments. This highlighted how their realization poses timing, scheduling, and resource management issues. Identified challenges include a lack of hardware support in virtualized environments (e.g., for timestamping), limitations of centralized offline scheduling in dynamic topologies, and the difficulty of achieving deterministic behavior in systems that inherently tend towards flexibility.

Several research directions must be pursued to fully leverage the benefits of TSN in virtualized networks. Firstly, ongoing research on developing accurate clock synchronization protocols should be directed towards fully software-based implementations. Moreover, flexible scheduling algorithms must be specifically designed and evaluated for dynamic environments while still enabling real-time guarantees. Furthermore, practical solutions for

solving the VNE problem in real time are needed. Further improving kernel-bypassing techniques could be a viable research direction to increase performance while preserving the configurability and scalability of virtualized networks.

References

[1] H. Chahed and A. Kassler, "Tsn network scheduling—challenges and approaches," *Network*, vol. 3, no. 4, pp. 585–624, 2023. [Online]. Available: https://www.mdpi.com/2673-8732/3/4/26

[2] A. Weder, "TIME SENSITIVE NETWORKING: Eine Einführung in TSN," Fraunhofer IPMS, Dresden, Germany, White Paper, 2019. [Online]. Available: http://s.fhg.de/time-sensitive-networking

[3] N. M. Mosharaf Kabir Chowdhury and R. Boutaba, "Network virtualization: state of the art and research challenges," *IEEE Communications Magazine*, vol. 47, no. 7, pp. 20–26, 2009.

[4] L. Bonati, M. Polese, S. D'Oro, S. Basagni, and T. Melodia, "Open, programmable, and virtualized 5g networks: State-of-the-art and the road ahead," *Computer Networks*, vol. 182, p. 107516, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128620311786

[5] M. F. Bari, R. Boutaba, R. Esteves, L. Z. Granville, M. Podlesny, M. G. Rabbani, Q. Zhang, and M. F. Zhani, "Data center network virtualization: A survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 2, pp. 909–928, 2013.

[6] M. Pahlevan, "Time sensitive networking for virtualized integrated real-time systems," Doctor of Engineering Dissertation, University of Siegen, Siegen, Germany, 2019.

[7] F. Rezabek, M. Bosk, G. Carle, and J. Ott, "Tsn experiments using cots hardware and open-source solutions: Lessons learned," in *2023 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, 2023, pp. 466–471.

[8] N. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Computer Networks*, vol. 54, no. 5, pp. 862–876, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128609003387

[9] A. Laghrissi and T. Taleb, "A survey on the placement of virtual resources and virtual network functions," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1409–1434, 2019.

[10] A. Garbugli, L. Rosa, A. Bujari, and L. Foschini, "Kubernetsn: a deterministic overlay network for time-sensitive containerized environments," in *ICC 2023 - IEEE International Conference on Communications*. IEEE, 2023, p. 1494–1499. [Online]. Available: http://dx.doi.org/10.1109/ICC45041.2023.10279214

[11] A. Garbugli, L. Rosa, L. Foschini, A. Corradi, and P. Bellavista, "A framework for tsn-enabled virtual environments for ultra-low latency 5g scenarios," in *ICC 2022 - IEEE International Conference on Communications*, 2022, pp. 5023–5028.

[12] M. Ashjaei, L. Lo Bello, M. Daneshtalab, G. Patti, S. Saponara, and S. Mubeen, "Time-sensitive networking in automotive embedded systems: State of the art and research opportunities," *Journal of Systems Architecture*, vol. 117, p. 102137, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1383762121001028

[13] J. Pei, Y. Hu, and L. Tian, "A review on key mechanisms of time-sensitive networking," in *2021 International Conference on Advanced Computing and Endogenous Security*, 2022, pp. 01–07.

[14] N. G. Nayak, F. Dürr, and K. Rothermel, "Incremental flow scheduling and routing in time-sensitive software-defined networks," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 5, pp. 2066–2075, 2018.

[15] K. Correll, N. Barendt, and M. Branicky, "Design considerations for software only implementations of the ieee 1588 precision time protocol."

[16] M. Pahlevan, N. Tabassam, and R. Obermaisser, "Heuristic list scheduler for time triggered traffic in time sensitive networks," vol. 16, no. 1, 2019. [Online]. Available: https://doi.org/10.1145/3314206.3314208

[17] C. Xue, T. Zhang, Y. Zhou, M. Nixon, A. Loveless, and S. Han, "Real-time scheduling for 802.1qbv time-sensitive networking (tsn): A systematic review and experimental study," in *2024 IEEE 30th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2024, pp. 108–121.

[18] J. Coleman, S. Almalih, A. Slota, and Y.-H. Lee, "Emerging cots architecture support for real-time tsn ethernet," in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. New York, NY, USA: Association for Computing Machinery, 2019, p. 258–265. [Online]. Available: https://doi.org/10.1145/3297280.3297542

[19] M. K. Hany, K. Montgomery, and R. Candell, "An analytical evaluation for software-based tsn in industrial wi-fi networks," in *2024 IEEE 7th International Conference on Industrial Cyber-Physical Systems (ICPS)*, 2024, pp. 1–6.

[20] R. Candell, K. Montgomery, M. Kashef Hany, S. Sudhakaran, and D. Cavalcanti, "Scheduling for time-critical applications utilizing tcp in software-based 802.1qbv wireless tsn," in *2023 IEEE 19th International Conference on Factory Communication Systems (WFCS)*, 2023, pp. 1–8.

[21] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013.

[22] O. Smimite and A. Karim, "Containers placement and migration on cloud system," *International Journal of Computer Applications*, vol. 176, 07 2020.

[23] J. Fried, G. I. Chaudhry, E. Saurez, E. Choukse, I. Goiri, S. Elnikety, R. Fonseca, and A. Belay, "Making kernel bypass practical for the cloud with junction," in *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. USENIX Association, Apr. 2024, pp. 55–73. [Online]. Available: https://www.usenix.org/conference/nsdi24/presentation/fried

[24] A. Francescon, G. Baggio, R. Fedrizzi, R. Ferrusy, I. G. Ben Yahiaz, and R. Riggio, "X–mano: Cross–domain management and orchestration of network services," in *2017 IEEE Conference on Network Softwarization (NetSoft)*, 2017, pp. 1–5.

[25] N. C. Fernandes, M. D. D. Moreira, I. M. Moraes, L. H. G. Ferraz, R. S. Couto, H. E. T. Carvalho, M. E. M. Campista, L. H. M. K. Costa, and O. C. M. B. Duarte, "Virtual networks: isolation, performance, and trends," *annals of telecommunications - annales des télécommunications*, vol. 66, no. 5, pp. 339–355, 2011.

[26] D. B. Oljira, "Low latency communication in virtualized and multipath networks," Ph.D. dissertation, 2020.

[27] P. Denzler, T. Frühwirth, C. Lehr, and J. Auffray, "Time-predictable software-based tsn-enabled network stack for mixed criticality traffic," in *2024 IEEE 20th International Conference on Factory Communication Systems (WFCS)*, 2024, pp. 1–8.

[28] A. Stage and T. Setzer, "Network-aware migration control and scheduling of differentiated virtual machine workloads," in *2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, 2009, pp. 9–14.

[29] Y. Zhang, Q. Xu, M. Li, C. Chen, and X. Guan, "Qos-aware mapping and scheduling for virtual network functions in industrial 5g-tsn network," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6.