Article

Utilization of SDN Technology for Flexible EtherCAT Networks Applications

Ireneusz Smółka * and Jacek Stój

Department of Distributed Systems and Informatic Devices, Faculty of Automatic Control, Electronics and Computer Science, Silesian University of Technology, 44-100 Gliwice, Poland; jacek.stoj@polsl.pl

* Correspondence: ireneusz.smolka@polsl.pl

Abstract: At the beginning of the current century, Ethernet-based communication networks began to be implemented in industrial applications. Some previously used protocols were migrated to Ethernet networks, while many others were strictly developed for this communication medium. Numerous industrial Ethernet protocols do not deliver all the capabilities provided by the Ethernet. For example, limitations may arise associated with wireless communication, use of dedicated switching devices, or operation solely for certain topologies. On the other hand, new technologies are now available, such as software defined networks (SDN), that add new features to Ethernet-based communication systems. In this paper, an EtherCAT network in combination with SDN is analyzed. EtherCAT network may only consist of devices with an implemented EtherCAT protocol stack. Therefore, regular Ethernet switches cannot typically be used in this network and, hence, special network infrastructure may be required to create topologies other than standard line topology. It is shown, however, that this limitation can be overcome by the application of SDN. In addition, a definition of datagram forwarding rules (called SDN flows here) is given, and we demonstrate that EtherCAT datagrams can be sent through routes that are required for proper EtherCAT network operation.

Keywords: software defined networks; networked control systems; real-time; industrial ethernet; EtherCAT; OpenFlow; OpenWRT

Citation: Smółka, I.; Stój, J. Utilization of SDN Technology for Flexible EtherCAT Networks Applications. *Sensors* **2022**, *22*, 1944. <https://doi.org/10.3390/s22051944>

Academic Editor: Xiaochun Cheng

Received: 30 November 2021

Accepted: 28 February 2022

Published: 2 March 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Efficient communication systems for industry are indispensable nowadays. New industrial systems are most often distributed, and they consist of many devices connected by networks. For many years, data exchange between nodes in industrial communication systems was based on fieldbuses. However, in the early 2000s, an Ethernet standard began to be extensively implemented for this purpose. Many protocols used for fieldbuses migrated to, after some adaptations, modern Ethernet communication links, e.g., Modbus TCP [1]. In addition, other protocols were developed strictly for the Ethernet, these include:

- Ethernet Powerlink EPL, introduced by the company B&R (Eggelsberg, Austria) in 2001.
- Ethernet/IP managed by ODVA (Open DeviceNet Vendors Association), launched in 2000.
- Profinet and EtherCAT, both presented for the first time at Hannover Fair in 2003.

Soon afterwards these protocols, alongside some others, became part of the international standard IEC 61158/IEC 61784-2 [2] and, today, they are among the most recognizable in the domain of industrial automation. It is noteworthy that another important protocol, which is not defined in this standard, is known as OPC UA; its first version was released in 2006 [3,4].

Even though numerous protocols were developed solely for application as an Ethernet standard, they cannot take advantage of all the characteristic features of an Ethernet network. The limitations start from the temporal requirements, typical in industrial systems

where operation in real-time is necessary. For example, in an EtherCAT network every device is a node in the network with the protocol implemented onto them. As a result, the usage of a common Ethernet infrastructure is not allowed in this case. The aim of this paper is to present a method to overcome this limitation.

EtherCAT communication is based upon the sending of consecutive EtherCAT datagrams from one EtherCAT slave device to another and then returning it to its origin, i.e., the EtherCAT master device [5]. Therefore, the communication is performed in a “logical ring”. Such communications can only be realized with a proper datagram flow—through EtherCAT nodes in a specific order. As a result, the usage of regular Ethernet switches is not possible because it would lead to an incorrect forwarding of the EtherCAT datagrams. A solution is to use software defined networks (SDN), which represents yet another type of virtualization that is already employed in computer systems [6].

In SDN, the forwarding of datagrams by switching devices accords to the programmed flows. Such switches allow the routing of the datagrams along specified paths; thus, EtherCAT communication through non-EtherCAT infrastructure becomes a possibility. This solution, called here as EtherCAT over SDN (EOS), is presented in this paper.

The latter is organized as follows. The following section presents previous works that relates communications in industrial systems to protocols based on the Ethernet, such as the EtherCAT. Section 3 describes the concept of SDN in application to EtherCAT-based systems. In Section 4, the experimental testbed is presented and the results from which are given in Section 5. Section 6 includes a discussion on these experimental results. The final section presents the conclusions based on our solution and offers future research possibilities.

2. Related Works

The operation of industrial computer systems is concurrent with production plant and factory floor processes. Therefore, computer systems must respond to events that occur in the processes in real-time, i.e., in a timely manner [7]. The early version of the Ethernet standard was difficult to introduce to industrial real-time systems. It was caused by the CSMA/CD media access mechanism which made delays in data transfer unpredictable [8]. In this so called “shared Ethernet” technique, real-time communication had to be implemented at the application level. An example of such a protocol is Ethernet PowerLink EPL, in which the producer–distributor–consumer mechanism is implemented [9]. Here, communication is coordinated by a predefined manager called the Managing Node MN (i.e., the distributor of the data), which decides when other network nodes can transmit (or produce) their data. When the transmitted data are broadcast, they may be read (consumed) from the network by the MN, as well as the other nodes since most of the network devices are both producers and consumers of data. Therefore, in EPL networks, each device is indistinguishable, and they are called Controlled Nodes (CNs). At any given point in time, only one datagram is transmitted in the network—either by the MN or one of the CNs. Therefore, the data flow is clearly describable in terms of time. In other words, the time needed to distribute a dataset in the network is easily calculated [10].

The later development of a “switched Ethernet”, and a full-duplex communication system, allowed for an easier introduction of Ethernet networks to industrial systems, since new protocols capable of real-time operation had been created. One of the many examples is Ethernet/IP, managed by ODVA, which combines the Ethernet standard and TCP/IP communication with CIP (Common Industrial Protocol) at the application level [11]. Here, data exchange is based on the provider-consumer principle. Yet, another example is the Profinet protocol [12], in which the communication model consists of three types of devices: an IO-Controller that coordinates communication, IO-Devices that exchange input–output (IO) data with the IO-Controller, and a IO-Supervisor that is required for network configuration and diagnostics (however, this is not needed during the normal operation of the network) [13]. The three aforementioned protocols are among the most popular for use in industrial applications. Similar to the EtherCAT protocol that is described in the next section.

New protocols continued to improve the applicability of the Ethernet standard to industrial applications. Alongside which the spatiotemporal description of the network operation and data flow has gained complexity. From this point of view, for a given industrial communication network that is known as the networked control system (NCS), the latency of data exchange should be calculated and analyzed. Previous research is available that compares the latency of aforementioned Ethernet protocols, such as Profinet and EtherCAT networks [14]. However, more recent works instead focused on the quality of the time synchronization. In an EtherCAT network, the latter is realized by use of Distributed Clock mechanisms, which can be improved by using system time drift compensation [15]. A solution, similar to those based on a Distributed Clock, has also been implemented on a Linux platform [16], which involved a time synchronization setup between Linux and the EtherCAT master [17].

Another area of research concerns industrial control systems (ICS) and the detection of anomalies in EtherCAT networks. Using machine learning techniques, the network traffic can be classified and various attack types flagged [18]. Anomaly detection may be also based upon device-level periodicity that can detect denial of service (DoS) and code-injection attacks [19]. Outside of cybersecurity, reliability is especially crucial in production systems. Ref. [20] describes a reliability enhancement method based on game theoretic feedback control. Of course, in order to improve the reliability of EtherCAT networks, a medium redundancy may also be applied by using a ring topology [21]. Other papers also mainly focus on EtherCAT applications; for example, in connection to the robot control of the EtherCAT master [22,23] or the design of the EtherCAT slaves [24].

In many protocols except EtherCAT (for example, Profinet), the communication network can be extensively similar to a typical Ethernet network. In such cases, common switches and wireless communication may be applied [25]. Moreover, for EPL and Ethernet/IP, there is no special hardware requirements and various topologies can be created. This type of flexibility is unavailable in EtherCAT networks because every component of such a network is required to be an EtherCAT network node and, thus, common switching devices cannot be used. Wireless communication is also unfeasible, except for higher level communication that use an EtherCAT Automation Protocol (EAP). Although an EtherCAT wireless extension [26] is available, which refers to acyclic communication that employs a Type 12 process data unit (PDU) and mailbox frames, it cannot be used in I/O data transfer.

The concepts presented in this paper could overcome these limitations by using SDN. It is shown that realization of EtherCAT communication using non-EtherCAT switches is obtainable. It means that, instead of using only line topology (which is typical for an EtherCAT network) or employing dedicated EtherCAT junction modules (for creation of topologies other than line ones), it is possible to create any topology using SDN switches. Thus, it makes the network structure more flexible and enables the possible ability to dynamically change the topology in order to connect new EtherCAT devices. Using SDN also creates another advantage, in that the sharing of the same network infrastructure for both EtherCAT and regular IT communication.

3. EtherCAT over SDN

In EtherCAT networks there are two types of communication methods: the EtherCAT Device Protocol (EDP) and the EtherCAT Automation Protocol (EAP). These are described below. The concept of software defined networks is then presented, followed by the description of the EtherCAT over SDN solution.

3.1. EtherCAT Protocol

Communication using EtherCAT Device Protocol, which is typically simply called the EtherCAT protocol, is based upon a master–slave principle. However, in contrast to other master–slave communication networks (for example, the well-known Modbus communication network or Profibus, in which the master–slave principle is applied alongside the token passing mechanism between the master stations), the EtherCAT master station does

not send separate datagrams that exchanges data with different slave stations using the request–response approach. EtherCAT datagrams contain processing data that propagates through all the EtherCAT slaves, which are called EtherCAT Slave Controllers ESC. This means that only one datagram is required for communicating with many ESC, which is then processed by the latter without the need for data buffering and storing. Therefore, minimal latency for the propagation of the datagram is introduced [20] and the exchange process for the input and output data becomes highly effective. In detail, when a datagram arrives at an input Ethernet port of a given ESC it is efficiently forwarded almost concurrently to its output port. The output dataset is extracted from the datagram “on-the-fly”; the input state is written similarly to the datagram.

The processing of the datagram is implemented by computer hardware in the form of a Fieldbus Memory Management Unit (FMMU) of the ESC. Datagrams traverse through FMMUs with a latency as low as 1 μ s for slave stations equipped with at least one Ethernet interface (100BaseTX or 100BaseFX) and 300 ns for slaves with only low voltage differential signal (LVDS) ports (this is also called an E-Bus interface, in devices by Beckhoff (Verl, Germany), which is used as a backplane connection in terminal modules such as input and output modules [27].) EtherCAT datagrams are transferred by just using Ethernet frames with broadcast MAC addresses. In other words, in reference to the ISO/OSI model, only the physical and data link layers (defined in Ethernet standard) are implemented; protocols from the upper layers are not used. In particular, no IP addressing is required since EtherCAT is only concerned with the application layers.

Apart from EDP, which is briefly described above, there is also the possibility of using EAP. This protocol involves communication between the master stations and it is based on the producer–consumer principle. However, the details on EAP are not described in this paper since we solely focus on EDP (for simplicity, this is just called EtherCAT later); the presented solution can, however, be applied to both. In EAP, “typical” Ethernet network devices and “typical” datagram processing can be used (but “on-the-fly” processing does not apply).

EtherCAT Device Protocol operates in a logical ring. Every datagram traverses from one ESC to another in a consecutive fashion. By taking advantage of full-duplex communication, the final topological ESC returns the EtherCAT datagrams to the master station, as shown in Figure 1. As stated earlier, it is not possible to include regular Ethernet switches in an EtherCAT network; one of the reasons is the need for broadcast MAC addressing. When a tree or star topology is built, a module called an EtherCAT junction should be used (also displayed in Figure 1).

Figure 1. Schematic showing the EtherCAT network, the topology for which is always a logical ring.

For the above reasons, common IT infrastructure cannot typically be utilized in EtherCAT communication, although its application could be made possible by using the Time Sensitive Network concept [28]. Since 2021, an IEEE task group has been attempting to standardize the real-time and safety of networks [29], and apply a critical enhancement for the Ethernet [30]. Unfortunately, we authors have not found any information on the

practical application of TSN in industrial scenarios. Hence, we present another solution based on SDN.

3.2. Software Defined Networks

In traditional Ethernet networks, the decisions on the routing of the datagrams are made by the switching devices. In SDN, an additional autonomous device is used that controls the data traffic—this is called the SDN controller. In SDNs, the control plane is separated from the data plane and the central SDN controller coordinates the data flow within the network. SDN only switches forward the datagrams according to the SDN controller instructions, which is known as the flows [31]. Great advantages may arise by decoupling the control and the data plane alongside a control of the centralized network. Above all, in such a scenario, decisions made by the SDN controller are based on a global view of the network. Therefore, all the decisions are more relevant to the current condition of the whole network [32].

SDN may be applied in various areas: Internet of Things [33] and Wireless Sensor Networks [34–36], industrial systems [37], or on the factory floor [38]. They can be implemented to improve the management and maintenance of networks [39], as well as increasing the security by use of Intrusion Detection Systems [26], protecting against eavesdropping attacks [40,41], creating resilient networks [42], and enhancing network reliability [43]; related papers also concern communication determinism [27]. In further reference to SDN, the Future Industrial Network Architecture (FIND) defines a guidance on the industry-related requirements in communication systems development [28]. For our purposes, SDN is applied so that datagrams forward to different ports of the SDN switching devices (called OVS or vSwitches). Moreover, every datagram is forwarded to the hosts of the networks, consecutively, in a predefined order. As a result, the topology resembles a line topology that corresponds to datagrams routing within EtherCAT networks.

3.3. EtherCAT over SDN

As stated earlier, a regular Ethernet infrastructure cannot be used within EtherCAT networks because a significant part of the datagrams in the EtherCAT is sent as broadcasts. Moreover, the broadcasted datagrams must reach every slave device in consecutive order. While typical Ethernet switches cannot cope with this type of datagram forwarding, the OVS within SDN are capable of this task. Despite the various addressing possibilities for the datagrams (i.e., either broadcasted, multicasted, or unicasted), forwarding by OVS is possible for any required scenario in a consecutive manner that accords to the needs of every slave device. By defining the appropriate flows in the SDN network, datagrams routing can be freely chosen according to, for example, the type, length, or payload of the datagrams.

4. Experimental Research

The concept of EtherCAT over SDN (also known as EOS) was verified during the experimental research. The testbed consisted of one Beckhoff controller (a EtherCAT master) and four ESC devices: EK1100 with some I/O modules. The EtherCAT network components were connected in two ways:

- A regular EtherCAT network in a line topology, which uses the switches that are integrated into the devices (see Figure 2).
- An implementation of the EOS concept, so that the ESC is connected to the master station via OVS (see Figure 3).

Figure 2. Schematic of the EtherCAT network, in a typical line topology configuration, used during the experimental research. The encircled digit “1” denotes the ET2000 probe location (measurements point no. 1).

Figure 3. Depiction of the EtherCAT over SDN (EOS) network solution in a star topology. The encircled digits “1” to “4” denote the ET2000 probe locations (measurements points no. 1 to 4).

In Figure 2, a typical EtherCAT network is presented that contains one master station and four ESC devices connected in a line topology. This is the most common topology in EtherCAT networks, since every ESC device contains an Ethernet switch built into the network interface. Other topologies are also possible, but they require the use of additional modules such as the EtherCAT extension modules (e.g., EK1110 by Beckhoff) or the EtherCAT junction modules (e.g., EK1122 by Beckhoff) that are mentioned in the previous section (see Figure 1). An identical set of EtherCAT devices, which are connected to an SDN network, representing the EtherCAT over SDN (EOS) solution is shown in Figure 3. For this scenario, any physical topology is possible; here, a star topology was chosen (as depicted in the figure) in order to present a topology that is significantly changed from the original line topology.

Using the EOS solution, a logical line topology that is typical for EtherCAT networks is realized by using an appropriate definition for the SDN flows. According to such flows, every datagram that is sent by the EtherCAT master is received on port P1.3 of the S1 OVS and then forwarded to port P1.4, which is where the ESC1 (remote I/O station) is connected. Every datagram that derives from the ESC1 slave on port P1.5 is forwarded to port P1.6, which is connected to the ESC2 slave, and so on. The datagrams eventually reach ESC4 on port P2.3, from which the ESC4 datagrams return to the master station via the same route (the opposite direction required for this is not marked on Figure 3). Of course, while the datagrams travel through every ESC device, the IO data associated with the given device

are processed, i.e., the output and input data are read from and written to the datagrams, respectively. This type of data processing is based on a “on-the-fly” approach, which means that the IO data can be processed even before the whole datagram is received from the network. The presented concept was verified during the experimental research; this is described in the next section. In greater depth, the main goal of the experimental research was to verify if EtherCAT communication can be coordinated in a SDN network, which was shown to be possible.

It is also important to determine the temporal costs that the OVS introduce to the EtherCAT communication network. Measurements on the network traffic were performed to obtain this necessity. For this purpose, an ET2000 multi-channel probe by Beckhoff was used. This enabled the capture of Ethernet datagrams with precise timestamping at the four monitoring points of the network; these are marked by encircled digits from “1” to “4” in Figure 3 referred latter as measurment points. The datagrams were registered using a regular PC with Wireshark software. Analysis of the timestamps enabled the measure of the network latencies, which are introduced by the devices located between the monitoring points. The timestamps are added, by the ET2000 probe, at the end of every Ethernet datagram. Therefore, the Ethernet traffic analysis is not dependent on the device used for capturing the datagrams. The timestamps had the resolution of 1 ns; however, the actual accuracy of the timestamping according to the vendor is 40 ns [44], whereas the delay in communication that the probe introduces to the network is lower than 1 μ s.

For the testbed illustrated in Figure 3, the following components were used:

- EtherCAT master: CX1020 embedded PC by Beckhoff;
- ESC: remote I/O station based on EK1100 EtherCAT ECS module with four EtherCAT I/O modules;
- SDN switches (OVS): virtual switch implementation based on Mikrotik RB2011UIAS-2HnD-IN;
- SDN controller: OpenDayLight controller installed on a PC with the Ubuntu operating system.

The two switches by Mikrotik (Riga, Litvia) that were used had their firmware changed to the OpenWRT and OpenvSwitch package. In addition, the L2 functionality of the switches was disabled. Therefore, without the defined SDN flows, the switches cannot provide any forwarding for the datagrams. Configuring the SDN flows required OpenFlow Manager, which was used to define the flows as described above. These flows did not contain any filtering, so that every datagram received on port P1.3 was forwarded to port P1.4, and so on.

5. Results

The Ethernet network traffic was captured for the two system topologies, i.e., with and without application of the EOS solution, as depicted by Figures 2 and 3. The results of which are presented in terms of the histograms of Figure 4 (for a system without SDN) and Figure 5 (with SDN and the EOS solution applied). In the experiment, 100,000 datagrams were captured and analyzed. In the former case, without the SDN, the x -axis represents time in the units of microseconds, while the y -axis is the number of measurements at a given network cycle, which was configured to be 10 ms. Most of the results reside at 10.1 ms, although significant numbers are seen at a slightly longer time. In general, the communication data remain within the configured cycle time with a 2% jitter. This shows that the latencies introduced by the ESC are minimal; the specifications of which are described in Section 3. Datagrams traverse through the FMMUs of the ESCs with latencies as low as 1 μ s for the slaves that contain at least one Ethernet interface (100BaseTX or 100BaseFX) and 300 ns for the slaves with only LVDS ports. The delay introduced by the ESC2-ESC4 and the module EK1100 of ESC1 equates to 1 μ s. While every I/O module of the ESC1 creates a latency of 300 ns. A total latency as little as 1.2 μ s, for the four modules, is thus highly plausible.

Figure 4. Histogram showing the update time of the EtherCAT ESC in a network without the SDN.

Figure 5. Histograms for the EtherCAT with SDN: (a) measurements taken at point no. 1, which provide the latency time for datagram routing through all the devices, and (b) measurements at points no. 1 through 4.

The system with the implemented EOS solution was more thoroughly tested. Measurements at the four different points were taken, which enabled a calculation of the EtherCAT datagrams flow in terms of latency time.

Figure 5a represents the variation in latency for the EtherCAT datagrams with the EOS solution implemented. Every measurement corresponds to the latency of a datagram that is transferred through the network. For example, over 16,000 datagrams each required a latency time of 460 μ s to be transferred from the PLC to the final ESC device and returned.

Here, in contrast to Figure 4, only the latencies are presented with no reference to the network cycle time (which was again set at 10 ms). For the EOS solution, compared to the system without SDN, the latency values are significantly greater and the distribution of them is much more spread out. The most likely measurement is 460 μ s and over 90% of the measurements reside in a range from 360 to 680 μ s—the jitter is, thus, much greater when the EOS solution is applied. This dataset is represented in the figure by ESC1^{RX}, which refers to the time period that begins when the datagrams are sent from the PLC (through the measurement point no. 1) and ends when the returned datagram is received from ESC1. In other words, the time required for the datagram to travel through the ESC1, ESC2, ESC3, and ESC4 devices and back. In an EtherCAT network, this involves one datagram that is consecutively processed by all the ESC.

The latency of ESC1^{RX} is calculated based on the datagrams captured with the ET2000 device at the measurement point no. 1. For every measurement point, the two Ethernet interfaces with the ET2000 probe are related. Datagrams that are received on one port are then forwarded to another port (which applies to every measurement port) and every pair of ports creates a monitoring channel. Additionally, the incoming datagrams are duplicated and sent to the monitoring port (in detail, the uplink—ninth Ethernet port of the ET2000 probe is used, while the ports of the monitoring points operate in the 100 Mbs mode and the uplink, i.e., the port with the duplicated datagrams, is a 1 Gbs port). The monitoring port is connected to the capturing device—namely, a PC with Wireshark software. The duplicated datagrams are extended by a status field (called the EtherCAT switch link), which consists of the port number that corresponds to when the datagram was received and a timestamp. For the measurement point no. 1, the calculations for the delay between receiving the datagrams from the PLC and the returned datagrams from ESC1 are determined. Other descriptions, similar to ESC1^{RX}, are displayed in Figure 5b. They refer to the following measurements:

- ESC1^{TX}—the latency between the datagrams received from the PLC at measurement points no. 1 and no. 2. It determines the time required for the datagrams to travel from point no. 1 to 2.
- ESC2^{TX}—the time needed for datagram transfer from point no. 1 to 3.
- ESC3^{TX}—as above, but between points no. 1 and no. 4.
- ESC4^{RX}—the latency between the datagrams received from the PLC at point no. 1 and the returned datagrams arriving at point no. 4. It determines the time need for datagram transfer from point no. 1 to point no. 4, which is added to the time taken from ESC4 back to point no. 4.
- ESC3^{RX}—as above but applies to datagram transfer through points from no. 1 to point no. 4, then ESC4 back to point no. 3.
- ESC2^{RX}—as above but with back transfer to point no. 2.
- ESC1^{RX}—the latency for the transfer of datagrams from point no. 1 to ESC4 and the return journey to point no. 1.

A summary of the mean and median of the measured results is presented in Table 1. The difference between the ESC1^{TX} and ESC2^{TX} values is very small, because it refers to the latency introduced by the ESC2 device only. According to its specifications it should be 1 μ s, which is consistent with the presented results. The variation between ESC2^{TX} and ESC3^{TX} corresponds to the latency of the OVS—which is around 85 μ s. However, the difference between ESC3^{TX} and ESC4^{RX} includes two passages through the OVS switches and one through the ESC4; here, the difference is around 100 μ s. Moreover, the value of ESC1^{TX} includes a double routing through the OVS S1 switch, which shows some discrepancy since it has an average value around 147 μ s. However, this inconsistency may result from not using a very powerful device, which is made from a regular OpenWRT switch, in the role of the OVS. It is expected that the results will be much more consistent when a dedicated OVS is used. Nevertheless, the experiment shows that the EOS solution can operate as required, although its temporal characteristics should be fully considered when applied to real-time systems. It is noteworthy that a similar OVS switch could be implemented on a Raspberry Pi device, which is equipped with Ethernet interfaces connected to the

USB ports. However, the latency introduced by this type of switch could be much more significant [45].

Table 1. A summary of the latency measurements (in units of μs) in terms of the average (the mean) and the median values.

	SDN (EOS Solution Implemented)						
	ESC1 ^{TX}	ESC2 ^{TX}	ESC3 ^{TX}	ESC4 ^{RX}	ESC3 ^{RX}	ESC2 ^{RX}	ESC1 ^{RX}
AVG	147.3	148.7	234.1	334.7	380.7	381.9	474.7
MEDIAN	133.9	135.1	222.0	319.3	363.9	365.1	455.6

6. Discussion

Our experimental research demonstrated that, by the implementation of SDN technology, it is possible for an appropriate routing of EtherCAT datagrams to create communication in EtherCAT networks via SDN. The definition of flows for the EtherCAT datagrams using OVS enable the ESC devices to operate seemingly in a line topology, while the actual (physical) architecture may not have this topological characteristic at all. As a result, the EtherCAT traffic can be routed through regular IT infrastructure, which enables efficient real-time interconnects between different areas of a production plant or factory floors without a need for elaborate and expensive devices.

Importantly, inclusion of additional devices in the EtherCAT network (i.e., the vSwitches/OVS) will introduce some extra latency to the datagram transfer. In the testbed experiment used, this was measured to be around 85 μs . On a comparison of this value with the latency of the EtherCAT slaves, which are as low as 1 μs or even 300 ns (depending on the type of slave device used), the OVS latency may seem to be overly large. Nevertheless, in most industrial applications, the network cycle is expressed in milliseconds rather than microseconds. Therefore, even with these types of delays, the presented solution can be effectively applied to real scenarios. Nevertheless, of course, it cannot be employed when high-precise communication is required, for example, in the synchronous operation of servo drives; in most cases, however, it should be applicable. In addition, the measured latency applies to the general-purpose switches (the Mikrotik) that were used in the testbed experiments, where the firmware is changed to OpenWRT. It is expected, therefore, that more advanced solutions that significantly minimize the latency in such cases should be possible.

Another drawback of the presented solution is that double Ethernet cables are necessary for the connection of every ESC to the OVS. Importantly, these additional cables are only required between an OVS and the first ESC, since the other ESC can be connected directly to the first ESC in a line topology. In future work, there are plans to eliminate this requirement, so that the operation of the OVS is similar to the EtherCAT junction module shown in Figure 1, in which only one Ethernet cable for every ESC is needed.

7. Conclusions and Future Work

In this study, we demonstrated that SDN technology can be implemented in the region of industrial Ethernet applications and it may overcome some of the limitations associated with EtherCAT networks. Typically, for example, EtherCAT networks are required to have a line topology when no EtherCAT junction modules are employed and regular Ethernet switches cannot usually be applied. We have demonstrated that both of these limitations can be removed by the use of SDN. Application of common switches, such as OVS (or SND switches), may deliver flexible network infrastructure even when industrial communication protocols such as EtherCAT are used. Moreover, common IT network infrastructure become useable in such networks if it supports SDN. Another advantage is that changes to the structure of the network is possible by a modification of the program, so that physical rewiring of the network can be avoided. To enable system architecture

to be even more flexible, future works plan to implement wireless communication in EtherCAT networks, since it should be possible to define SDN flows that enable the routing of EtherCAT datagrams in a Wi-Fi network. The general intention, for this future research, is to interconnect more than two devices with one wireless network.

The presented solution is applicable not only to EtherCAT networks but also systems based on other protocols. For example, in EPL networks, only one distributor node may be used, called the Managing Node (MN), which creates one real-time domain. When additional MNs are present, separate Ethernet infrastructure should, in general, be used. Moreover, different EPL real-time subsystems may coexist on the same infrastructure, which has applications to managed Ethernet switches and the logical separation of the domains by a VLANs configuration. However, such domain separation could be based on SDN in future. It would increase the security of the EPL by not forwarding datagrams that arrive from other ports of the OVS (SDN vSwitches), which is where the controlled nodes (CN) are connected; for example, those from outside the processing network.

Author Contributions: Conceptualization, I.S. and J.S.; methodology, I.S. and J.S.; software, I.S.; validation, I.S. and J.S.; formal analysis, I.S. and J.S.; writing—original draft preparation, J.S.; writing—review and editing, I.S. and J.S.; visualization, J.S.; funding acquisition, I.S. All authors have read and agreed to the published version of the manuscript.

Funding: The work reported in the paper is partially supported by the European Social Funds; project no POWR.03.02.00-00-I007/17-00 “CyPhIS—the program of modern PhD studies in the field of Cyber-Physical Systems”. The research leading to these results received funding from the Norway Grants 2014–2021, which the National Centre operates for Research and Development under the project “Automated Guided Vehicles integrated with Collaborative Robots for Smart Industry Perspective” (Project Contract no.: NOR/POLNOR/CoBotAGV/0027/2019-00) and partially by the Polish Ministry of Science and Higher Education Funds for Statutory Research. Publication supported by the Own Scholarship Fund of the Silesian University of Technology in the year 2019/2020, grant number: 22/FSW18/0003-03/2019. This work was partially supported by Statutory Research funds of the Department of Distributed Systems and Informatic Devices, Silesian University of Technology, Gliwice.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

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

References

1. Dong, L.; Yan, D. Design and Implementation of Energy Consumption Data Acquisition System of Tram Based on Lab View. In Proceedings of the 2021 IEEE 4th International Conference on Computer and Communication Engineering Technology (CCET), Beijing, China, 13–15 August 2021; pp. 420–424.
2. Akima, T.; Shibata, K. Development of real-time ethernet based I/O network. In Proceedings of the 2008 SICE Annual Conference, Chofu, Japan, 20–22 August 2008; pp. 83–86.
3. Cupek, R.; Ziebinski, A.; Drewniak, M. An OPC UA server as a gateway that shares CAN network data and engineering knowledge. In Proceedings of the 2017 IEEE International Conference on Industrial Technology (ICIT), Toronto, ON, Canada, 22–25 March 2017; pp. 1424–1429.
4. Cupek, R.; Ziębiński, A.; Drewniak, M.; Fojcik, M. Application of OPC UA Protocol for the Internet of Vehicles. In *Proceedings of the Computational Collective Intelligence*; Nguyen, N.T., Papadopoulos, G.A., Jędrzejowicz, P., Trawiński, B., Vossen, G., Eds.; Springer International Publishing: Berlin/Heidelberg, Germany, 2017; pp. 272–281.
5. Stój, J. Cost-Effective Hot-Standby Redundancy with Synchronization Using EtherCAT and Real-Time Ethernet Protocols. *IEEE Trans. Autom. Sci. Eng.* **2020**, *18*, 2035–2047. [\[CrossRef\]](#)
6. Gaj, P.; Skrzewski, M.; Stój, J.; Flak, J. Virtualization as a way to distribute PC-based functionalities. *IEEE Trans. Ind. Inform.* **2015**, *11*, 763–770. [\[CrossRef\]](#)
7. Scanzio, S.; Wisniewski, L.; Gaj, P. Heterogeneous and dependable networks in industry—A survey. *Comput. Ind.* **2021**, *125*, 103388. [\[CrossRef\]](#)
8. Domański, A.; Domańska, J.; Czachórski, T.; Klamka, J.; Szyguła, J.; Marek, D. The IoT gateway with active queue management. *Int. J. Appl. Math. Comput. Sci.* **2021**, *31*, 165–178. [\[CrossRef\]](#)

9. POWERLINK Basics–Brochure. Available online: https://www.ethernet-powerlink.org/uploads/media/POWERLINKBasics_brochure_e.pdf (accessed on 5 October 2021).
10. Romanov, A.; Slepynina, E. Real-time Ethernet POWERLINK Communication for ROS. Part I. General Concept. In Proceedings of the 2020 Ural Smart Energy Conference (USEC), Ekaterinburg, Russia, 13–15 November 2020; pp. 159–162.
11. Niu, C.; Xie, Z.; Zhou, T. Application of Ethernet/IP Protocol in Equipment Lifecycle Management. In Proceedings of the 2021 IEEE International Conference on Information Communication and Software Engineering (ICICSE), Chengdu, China, 19–21 March 2021; pp. 260–263.
12. Kampen, A.-L.; Fojcik, M.; Cupek, R.; Stoj, J. The requirements for using wireless networks with AGV communication in an industry environment. In Proceedings of the 2021 17th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Bologna, Italy, 11–13 October 2021; pp. 212–218.
13. Dias, A.L.; Sestito, G.S.; Turcato, A.C.; Brandão, D. Panorama, challenges and opportunities in PROFINET protocol research. In Proceedings of the 2018 13th IEEE International Conference on Industry Applications (INDUSCON), Sao Paulo, Brazil, 12–14 November 2018; pp. 186–193.
14. Wu, X.; Xie, L.; Lim, F. Network delay analysis of EtherCAT and PROFINET IRT protocols. In Proceedings of the IECON 2014–40th Annual Conference of the IEEE Industrial Electronics Society, Dallas, TX, USA, 29 October–1 November 2014; pp. 2597–2603.
15. Park, S.-M.; Kim, H.; Kim, H.-W.; Cho, C.N.; Choi, J.-Y. Synchronization Improvement of Distributed Clocks in EtherCAT Networks. *IEEE Commun. Lett.* **2017**, *21*, 1277–1280. [[CrossRef](#)]
16. Park, S.-M.; Kim, H.-W.; Kim, H.-J.; Choi, J.-Y. Accuracy Improvement of Master–Slave Synchronization in EtherCAT Networks. *IEEE Access* **2020**, *8*, 58620–58628. [[CrossRef](#)]
17. Park, S.-M.; Kwon, Y.; Choi, J.-Y. Time Synchronization Between EtherCAT Network and External Processor. *IEEE Commun. Lett.* **2021**, *25*, 103–107. [[CrossRef](#)]
18. Akpinar, K.O.; Ozcelik, I. Analysis of Machine Learning Methods in EtherCAT-Based Anomaly Detection. *IEEE Access* **2019**, *7*, 184365–184374. [[CrossRef](#)]
19. Akpinar, K.O.; Ozcelik, I. Methodology to Determine the Device-Level Periodicity for Anomaly Detection in EtherCAT-Based Industrial Control Network. *IEEE Trans. Netw. Serv. Manag.* **2021**, *18*, 2308–2319. [[CrossRef](#)]
20. Li, L.; Cong, P.; Cao, K.; Zhou, J.; Wei, T.; Chen, M.; Hu, S.; Hu, X.S. Game Theoretic Feedback Control for Reliability Enhancement of EtherCAT-Based Networked Systems. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2019**, *38*, 1599–1610. [[CrossRef](#)]
21. Zhang, J.; Wang, B.; Liu, J.; Zhang, L. PWM Precise Synchronous Control Method Based on EtherCAT Distributed Ring Net Topology. In Proceedings of the 2020 39th Chinese Control Conference (CCC), Shenyang, China, 27–29 July 2020; pp. 4487–4491.
22. Delgado, R.; Kim, S.; You, B.; Choi, B. An EtherCAT-based real-time motion control system in mobile robot application. In Proceedings of the 2016 13th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), Xi'an, China, 19–22 August 2016; pp. 710–715.
23. Chuang, W.-L.; Yeh, M.-H.; Yeh, Y.-L. Develop Real-Time Robot Control Architecture Using Robot Operating System and EtherCAT. *Actuators* **2021**, *10*, 141. [[CrossRef](#)]
24. Robertz, S.G.; Henriksson, R.; Nilsson, K.; Blomdell, A.; Tarasov, I. Using Real-Time Java for Industrial Robot Control. In *Proceedings of the 5th International Workshop on Java Technologies for Real-Time and Embedded Systems*; Association for Computing Machinery: New York, NY, USA, 2007; pp. 104–110.
25. Gaj, P.; Maćkowski, M. Electromagnetic compatibility issues in hybrid wired and wireless industrial networks. *PLoS ONE* **2020**, *15*, e0232405. [[CrossRef](#)] [[PubMed](#)]
26. Wu, X.; Xie, L. On the Wireless Extension of PROFINET Networks. In Proceedings of the 2019 IEEE VTS Asia Pacific Wireless Communications Symposium (APWCS), Singapore, 28–30 August 2019; pp. 1–5.
27. ETG.1600G (R)V1.0.2 EtherCAT Installation Guideline: Guideline for Planning, Assembling and Commissioning of EtherCAT Networks. EtherCAT Technology Group. Available online: <https://documents.pub/document/etg1600-ethercat-installation-guideline-installation-guideline-guideline-for.html> (accessed on 9 August 2017).
28. EtherCAT Technology Group. *ETG.1700S(D)V0.9.1 EtherCAT TSN Communication Profile*; EtherCAT Technology Group: Nürnberg, Germany, 2018.
29. Peserico, G.; Morato, A.; Tramarin, F.; Vitturi, S. Functional Safety Networks and Protocols in the Industrial Internet of Things Era. *Sensors* **2021**, *21*, 6073. [[CrossRef](#)] [[PubMed](#)]
30. Gavriluț, V.; Pop, P. Traffic-type Assignment for TSN-based Mixed-criticality Cyber-physical Systems. *ACM Trans. Cyber-Phys. Syst.* **2020**, *4*, 23. [[CrossRef](#)]
31. SDN (Software-Defined Networking). In *Software Networks*; John Wiley & Sons, Ltd.: Hoboken, NJ, USA, 2020; pp. 13–32. ISBN 978-1-119-69474-8.
32. Imran, Ghaffar, Z.; Alshahrani, A.; Fayaz, M.; Alghamdi, A.M.; Gwak, J. A Topical Review on Machine Learning, Software Defined Networking, Internet of Things Applications: Research Limitations and Challenges. *Electronics* **2021**, *10*, 880. [[CrossRef](#)]
33. Urrea, C.; Benítez, D. Software-Defined Networking Solutions, Architecture and Controllers for the Industrial Internet of Things: A Review. *Sensors* **2021**, *21*, 6585. [[CrossRef](#)] [[PubMed](#)]
34. Yoon, H.; Kim, S.; Nam, T.; Kim, J. Dynamic flow steering for IoT monitoring data in SDN-coordinated IoT-Cloud services. In Proceedings of the 2017 International Conference on Information Networking (ICOIN), Da Nang, Vietnam, 11–13 January 2017; pp. 625–627.

35. Tomovic, S.; Radusinovic, I. Performance analysis of a new SDN-based WSN architecture. In Proceedings of the 2015 23rd Telecommunications Forum Telfor (^{TEL}FOR), Belgrade, Serbia, 24–26 November 2015; pp. 99–102.
36. Puente Fernández, J.A.; García Villalba, L.J.; Kim, T.-H. Software Defined Networks in Wireless Sensor Architectures. *Entropy* **2018**, *20*, 225. [[CrossRef](#)]
37. Henneke, D.; Wisniewski, L.; Jasperneite, J. Analysis of realizing a future industrial network by means of Software-Defined Networking (SDN). In Proceedings of the 2016 IEEE World Conference on Factory Communication Systems (WFCS), Aveiro, Portugal, 3–6 March 2016; pp. 1–4.
38. Ahmed, K.; Blech, J.O.; Gregory, M.A.; Schmidt, H. Software Defined Networking for Communication and Control of Cyber-Physical Systems. In Proceedings of the 2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS), Melbourne, VIC, Australia, 14–17 December 2015; pp. 803–808.
39. Wazirali, R.; Ahmad, R.; Alhiyari, S. SDN-OpenFlow Topology Discovery: An Overview of Performance Issues. *Appl. Sci.* **2021**, *11*, 6999. [[CrossRef](#)]
40. Ndonga, G.K.; Sadre, R. A low-delay SDN-based countermeasure to eavesdropping attacks in industrial control systems. In Proceedings of the 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Berlin, Germany, 6–8 November 2017; pp. 1–7.
41. Ujjan, R.M.A.; Pervez, Z.; Dahal, K.; Khan, W.A.; Khattak, A.M.; Hayat, B. Entropy Based Features Distribution for Anti-DDoS Model in SDN. *Sustainability* **2021**, *13*, 1522. [[CrossRef](#)]
42. Josbert, N.N.; Ping, W.; Wei, M.; Li, Y. Industrial Networks Driven by SDN Technology for Dynamic Fast Resilience. *Information* **2021**, *12*, 420. [[CrossRef](#)]
43. Shi, Y.; Zhang, Y.; Jacobsen, H.-A.; Tang, L.; Elliott, G.; Zhang, G.; Chen, X.; Chen, J. Using Machine Learning to Provide Reliable Differentiated Services for IoT in SDN-Like Publish/Subscribe Middleware. *Sensors* **2019**, *19*, 1449. [[CrossRef](#)] [[PubMed](#)]
44. ET2000 Industrial Ethernet Multichannel Probe. Beckhoff. Available online: <https://www.beckhoff.com/en-en/products/i-o/ethercat-development-products/elxxxx-etxxxx-fbxxxx-hardware/et2000.html> (accessed on 18 January 2016).
45. Stój, J.; Ziębiński, A.; Cupek, R. FPGA based Industrial Ethernet Network Analyser for Real-time Systems Providing Openness for Industry 4.0. *Enterp. Inf. Syst.* **2021**, 1–21. [[CrossRef](#)]