Fragmentation in Reconfigured Real-time Production Networks

Christian von Arnim, Armin Lechler, Oliver Riedel, and Alexander Verl

University of Stuttgart, Institute for Control Engineering of Machine Tools and
Manufacturing Units, D-70174 Stuttgart, Germany

christian.von-arnim@isw.uni-stuttgart.de

<https://www.isw.uni-stuttgart.de/>

Abstract. A reconfigurable production is able to adapt to a new product in a short time. This can be built up from cyber-physical production systems (CPPS) which exchange various data with several devices simultaneously via a convergent network. In these future production systems, convergent networks will connect more participants than real-time production networks do today. In order to enable uninterrupted operation, reconfigurations must not affect existing communication. Due to this limitation, fragmentation of the configuration can occur, restricting future operations. This paper investigates fragmentation caused by reconfigurations using a simulation to determine whether fragmentation can lead to problems in future production networks.

Keywords: real-time production network, fragmentation, reconfiguration

1 Introduction

The industry 4.0 production must be able to produce a wide range of variants up to batch size one and at the same time execute the production in the shortest possible time [1]. In order to achieve this, [1] and [15] propose a division into cyber-physical production systems (CPPS) consisting of a set of sensors and actuators integrated into a network. In the context of assembly, this corresponds to a switch from inflexible conveyor belts to automated guided vehicles (AGVs) [2]. CPPSs enable a reconfigurable manufacturing system that produces different products and can be adapted within a shorter period [16] instead of a flexible manufacturing system that can produce different variants of a product.

This concept does no longer fit into the automation pyramid, with clearly divided field and control level for each manufacturing system [4, 11]. Real-time communication occurs only at the lowest levels in locally limited real-time production networks (RTPNs) (e.g. within a machine), whereby sensors and actuators communicate with a single controller. The concept of levels dissolves during the transition to a CPPS-based production environment. The communication across levels increases so that, for example, an AGV independently accepts the next available production order. In parallel, the values of sensors are not only

used for control, but are also stored and evaluated by applications, for example in order to improve the early detection of errors. Instead of separation into levels, this results in a network of different applications that communicate with each other. Real-time communication is no longer limited to local networks and spans the entire network from CPPS instead, as shown in Fig. 1 [15].

Fig. 1: The automation pyramid dissolves and separated RTPNs turn into one large RTPN. Adapted from [15].

Two possible network architectures that realize a CPPS-based automation are explained in [7] and [8]. Both topologies integrate all participants into one RTPN, which can be implemented with the vendor-independent time-sensitive networking (TSN) extensions to the IEEE 802.1 standard, for example [10]. The control is done in local cloud platforms, which execute applications for control. [8] also explains that this control can be executed on external cloud platforms. The network architecture is shown in Fig. 2.

Fig. 2: Network architectures in CPPS-based production systems.

The main difference to the previous architecture of different, isolated RTPNs is that all network participants are contained in a shared RTPN. This results in a RTPN with more participants than before, which can span an entire shop floor. Due to the local separation of the sensors/actuators and the controller,

the data must be transmitted over several switches, which must be configured in such a way that all real-time critical data reach their destination on time.

Due to its increased number of devices and new flexibility, changes may occur more frequently to this RTPN that require a new configuration of the switches. Examples for changes are the addition or removal of devices from the network. Furthermore, when starting a new application, such as real-time analysis of manufacturing data, additional data may need to be exchanged between the sensors/actuators and the execution platform. This must not affect the current communication. So, the rest of the production is not interrupted by the reconfiguration. In contrast, [9] only takes care during reconfiguration so that as few packets as possible fail and a modification of existing communication is tolerated.

The paper is structured as follows: In Section 2, RTPNs with minimum guaranteed latencies are introduced, followed by a description of the reconfiguration of these networks without interruption of the existing communication and the possibility of fragmentation in Section 3. In Section 4, a simulation of fragmentation is done and evaluated. Afterwards, a summary is given and possible further work is explained in Section 5 and Section 6.

2 Real-time Ethernet

RTPNs require minimal latency and jitter, which can be achieved by transmission at specified time slots. In addition, the CPPS architecture requires support for multiple senders and receivers. This can be accomplished, for example, by using TSN with 802.1Qbv [6]. It is assumed that all participants have a synchronized time, for example through the Precision Time Protocol (PTP) [5]. Within TSN, data are classified into traffic classes. The real-time capability is realized by defining that only data of a certain traffic class are transmitted in a time slot. Data of other traffic classes are not transmitted in this time slot. The communication runs cyclically, whereby the synchronized clocks ensure that all participants use the same synchronized cycle. This cycle is divided into time slots and provides an exclusive transmission path for the data of a traffic class. The division can be defined per port at the switch and does not have to be identical for the entire network.

In Fig. 3 this distribution is visualized in an example. A cycle from t to $t + 1$ is shown. From t to $t + t_1$ only data of the traffic class 1 can be sent, from $t + t_1$ to $t + t_2$ only traffic class 2 can be sent. The same applies to the third slot. Finally, there is a slot from $t + t_3$ to $t + 1$ which allows to send all data. This cycle is repeated continuously.

In the following, a common cycle time is assumed for the entire network. In case of different transmission cycles a hypercycle with cycle time of the least common multiple of all cycle times can be considered equivalent [3]. Also, only the traffic sent in a defined time slot is considered as real-time traffic in this article. Prioritized traffic, which is sent only prioritized to other traffic, is not considered separately and is seen as part of non-real-time traffic.

Fig. 3: A cyclic transmission with reserved time slots (1 to 3) guarantees that only traffic of specific class is transmitted within this period.

3 Reconfiguration of real-time production networks

As mentioned in Section 1, due to the larger RTPN and new possibilities to start/stop applications that access real-time data across the network, changes in the configuration of the RTPN are more likely. A reconfiguration adds and/or removes any number of time slots for real-time data. Existing time slots for real-time data cannot be modified (e.g. moved or split), as otherwise guarantees regarding latency and jitter could be violated [12]. This section points out which conditions must be met to reconfigure a network without disturbance to the existing real-time traffic. Afterwards, the problem of fragmentation is addressed as a consequence of the reconfiguration.

3.1 Execution of a reconfiguration in a real-time production network

When executing a reconfiguration that is not intended to restrict the functionality of the entire network, the possibilities of reconfiguration are limited in contrast to a fresh configuration without existing time slots. Time slots without real-time data do not have any requirements. These can be moved at will, shortened by time slots with real-time data (inserting a time slot at the edge) or interrupted (inserting a time slot in the middle).

In general, changing the configuration can be done in two steps. First, all time slots that are no longer required are removed from the current configuration. Afterwards, the newly added time slots are inserted into the available time ranges in the cycle that are not yet occupied with real-time data. Possible incremental algorithms that support the placement of new time slots are, for example, [12] and [13].

3.2 Fragmentation

When removing and inserting time slots of different sizes, gaps may occur between time slots because existing time slots remain in the cycle at their previous time during reconfiguration. Within the context of RTPNs we define fragmentation in a communication cycle when the reserved time slots have gaps between each other so that the remaining free space in the communication cycle is not available as a single block. This can cause problems when adding a larger time slot and there is no sufficiently large gap available even though the total sum of

time slots is less than the cycle time. This is similar to external fragmentation of storage systems, when files have been removed and added over time causing separated blocks of free memory [14]. However, the strategies to reduce external fragmentation (e.g. compaction) on storage systems can not be applied to fragmentation in RTPNs, as moving existing time slots within a cycle may violate guaranteed latencies. Fragmentation in the context of RTPNs is not the same as IP fragmentation, when IP packages are splitted during transmission.

An example of fragmentation is shown in Fig. 4, which visualizes a sequence of reconfigurations. At the beginning (step 1), there are five time slots, which occupy the whole cycle. Then, the time slots 1 and 4 are removed and the smaller slots 6 and 7 are inserted at their position (step 3). After removing the time slot 3, there are gaps between all existing time slots in the resulting cycle so that time slot 8 cannot be inserted although there is enough space in total.

Fig. 4: Example for fragmentation during reconfiguration. The time slot 8 can not be scheduled, although the total space is enough.

This demonstrates how fragmentation can cause problems over time, as small gaps between time slots cannot be used for larger time slots. To what extent and after how many reconfigurations this problem occurs is evaluated in the following using a simulation.

4 Simulation

In the simulation, fragmentation is evaluated based on a real-time connection between two network participants by removing and inserting multiple time slots of different sizes. It is not considered that the time slot in a network must be reserved on the complete path from the sender to the receiver. Fig. 5 shows this conflict. A time slot for data from switch A over B to C must be available in both $A \rightarrow B$ and $B \rightarrow C$. On the $A \rightarrow B$ route and on the $B \rightarrow C$ route there is enough space. However, on $A \rightarrow B$ the new time slot can only be inserted in the middle, while on $B \rightarrow C$ it can only be inserted at the end. Consequently, there is no time for the new slot that is available on both $A \rightarrow B$ and $B \rightarrow C$.

Since a solution for scheduling in networks is NP-hard [12] and different algorithms produce different scheduling results, only a single connection is con-

Fig. 5: A time slot across multiple switches requires free space on each connection at the same time. (Transmission time expected to be zero)

sidered in the simulation (e.g. the $A \rightarrow B$ connection). With this simplification, the results are expected to be better than in a network with multiple switches.

The simulation considers a cycle that is reconfigured several times. For the simulation a 100 Mbit network equivalent to current field buses is assumed, which exchanges data with a frequency of 1 kHz. Thereby, 12500 Bytes can be transmitted per cycle including the overhead of the protocol (e.g. Ethernet header). Below is a list of all the parameters of the simulation, some of which have been chosen identically to enable a comparison:

- Number of time slots (X)
- Number of time slots that are removed and inserted each iteration (Y)
- Minimum and maximum size of time slots
- Maximum number of iteration. Here: 10000
- Total size of the cycle. Here: 12500 Bytes

The simulated reconfiguration sequence is shown in Algorithm 1. Each time a new time slot is added, its size is randomly determined from a parameterized interval. During initialization, X time slots are inserted into a cycle whose size for this simulation is set to 12500Bytes. The time slots are all inserted into the cycle one after another so that there are no gaps between them (as shown in Fig. 5 in $B \rightarrow C$ with four time slots). Afterwards, the following steps are repeated: First, Y random time slots are removed. Then, Y random time slots are inserted into the existing gaps again. If several gaps are large enough for a time slot, the smallest gap is selected in order to have larger gaps available for any following, larger time slots. The two steps of removing and adding time slots are performed until either the time slots can no longer be inserted or the maximum number of iterations has been reached. Up to 10000 iterations are executed in this contribution.

```

Insert  $X$  time slots next to each other;
for  $i = 1$  to 10000 do
    Remove  $Y$  random time slots;
    Try to insert  $Y$  random time slots with random sizes, each time slot
    uses the smallest available gap;
    if Insert failed then
        break;

```

Algorithm 1: Simulation of reconfigurations.

For each parameter set (P) listed in Table 1, 1000 runs with different random sized time slots are performed. Each run contains up to 10000 reconfiguration iterations. The first parameter set (P_1) of the simulation uses 40 time slots (X) of a size of [249 : 250] and replaces one time slot per round (Y) so that the interval of 12500 is occupied between 79.68 % and 80.32 % in total. The result is shown as a histogram in Fig. 6. The histogram uses a bucket size of 100 so that the blue bar on the left represents the 133 simulations that failed within the first 100 iterations. The red line represents the total percentage of simulations that failed until an iteration, e.g. at 10000 it has a value of 93.5 %. Thus, 935 from 1000 runs of P_1 failed before iteration 10000.

As a result, only 65 runs out of 1000 were capable to process 10000 iterations, while 25 % of the runs were not able to reach 200 iterations. Transferred to production, this means that in case of this parameterization, in 25 % of the cases no 200 reconfigurations of the network can be performed without the requirement of moving previous scheduled time slots. But depending on the initial configuration and on the required changes, it might also be possible to reach over 10000 reconfigurations.

Fig. 6: Histogram of 1000 runs of parameter set P_1 .

Table 1 lists all simulations performed with parameters and results. All simulations have in common that they use 80 % of the available cycle (1000 of 12500) in average. The first column is the number of the parameter set. The next three columns contain the parameters used in that simulation, the number of removed and inserted time slots per iteration (Y), the range of time slot sizes and the total number of time slots (X).

The other columns show evaluated results of the 1000 runs per simulation. "Min./max. iterations" is the minimum and maximum number of iterations. A value of ≥ 10000 means that the simulation has stopped after 10000 iterations. "Failed" contains the count of the 1000 runs in total that do not reach the 10000 iterations, "passed" contains the runs that reach 10000 iterations. The last two columns show the iteration when 25 %/90 % of the runs have failed. When less

than 25 %/90 % of the simulation runs failed in total, there is no value ("–"). In Fig. 6 this value is the number of iterations when the red line reaches 25 %/90 %.

Table 1: Parameters and results of the simulation

	Y	time slot sizes	X	min. iterations	max. iterations	failed	passed	25 %	90 %
P_1	1	[249, 251]	40	13	≥ 10000	935	65	172	7900
P_2	1	[248, 252]	40	16	≥ 10000	998	2	195	2847
P_3	1	[247, 253]	40	19	7200	1000	0	154	1986
P_4	1	[246, 254]	40	21	5434	1000	0	148	1633
P_5	1	[245, 255]	40	22	4059	1000	0	123	1414
P_6	1	[124, 126]	80	128	≥ 10000	11	989	-	-
P_7	1	[84, 86]	118	339	≥ 10000	1	999	-	-
P_8	1	[499, 501]	20	8	1465	1000	0	33	290
P_9	2	[249, 251]	40	14	≥ 10000	965	35	152	6888
P_{10}	3	[249, 251]	40	8	≥ 10000	953	47	293	6756
P_{11}	4	[249, 251]	40	6	≥ 10000	950	50	528	7804
P_{12}	5	[249, 251]	40	6	≥ 10000	930	70	662	8752
P_{13}	6	[249, 251]	40	5	≥ 10000	904	96	882	9896

Regarding the simulations P_1 to P_5 , the maximum number of iterations decreases from ≥ 10000 to 4059 when the size of time slots varies more ([249, 251] to [245, 255]). So, wider range of time slot sizes can therefore reduce the number of successful reconfigurations. This can be explained by the strategy that the smallest possible gap is used when inserting a time slot. When the largest time slot should be inserted, in P_1 , only two other time slot sizes ([249, 250]) might occupy a previous removed largest time slot, while in P_5 ten smaller time slots ([245, 254]) can occupy a largest time slot. Thus, the chance of finding a matching gap between two time slots is the smaller the more possible time slot sizes are available.

Using smaller but more time slots in P_6 and P_7 , compared to P_1 , fewer runs (11 in P_6 and 1 in P_7) failed within 10000 iterations. One explanation for this is that more iterations are needed to fill 20 % of the available space if no smaller gap is found for an inserted time slot, as the time slot is smaller. By doubling the size of the time slots (P_8 compared to P_1), the maximum number of iterations decreases from ≥ 10000 to 1465.

Comparing P_1 with P_9 to P_{13} , the amount of passed runs varies within a range of [35, 96]. In P_{10} to P_{13} the number of iterations till 25 % percent of the runs fail is higher than P_1 . So, increasing the number of changed time slots per reconfiguration (Y) from one to two firstly causes more failures. From 3 on (P_{10} to P_{13}) more reconfigurations are possible before 25 % of the runs failed. Two effects might cause this behavior. At first, the first iteration may remove only time slots of the minimum size and only inserts slots with a greater size so that all inserted slots must be placed at the end of the previously inserted time slots. When this is done for more slots simultaneously, then more gaps are

in the cycle and the buffer at the end becomes smaller more quickly. Otherwise, the more time slots are reconfigured, the more likely it is that the size of some removed time slots will match that of some time slots to be inserted. In case of 40 reconfigured time slots per iteration, all time slots are removed and inserted again, so there will be no fragmentation at all.

In summary, the simulation of a single wire between two switches identifies the following behaviors:

- The greater the amount of sizes, the lower the average number of successful reconfigurations.
- Smaller time slots enable more reconfigurations than larger time slots.
- The reconfiguration of multiple time slots at once has a positive effect on the number of reconfigurations, after a deterioration at the beginning.

5 Summary

In a reconfigurable production based on CPPS, new network architectures with larger RTPNs are used. The increased number of participants and the common reconfiguration of the production lead to an increased number of reconfigurations of the RTPN in operation, whereby parts of the configuration must be exchanged without affecting the existing communication. If real-time communication is implemented as a cyclic data transport, in which fixed time slots are reserved (e.g. using IEEE802.1Qbv), the reconfiguration cannot move existing time slots. This can lead to fragmentation when time slots are added and deleted during reconfiguration. Within this paper a simulation has been used to check how often a network can be reconfigured for specific sets of parameters. Depending on the parameters, a RTPN can only be reconfigured a few hundred times before gaps between existing time slots prevent a successful reconfiguration.

6 Further work

On hard disks, the problem of fragmentation has been solved by running defragmentation operations that move existing files on the disk. To solve the problem of fragmentation in RTPNs equivalent concepts need to be developed and methods for modifying existing time slots during a reconfiguration are required. One solution is that some applications can be customized to tolerate scheduling the time slot to a new position during runtime within a certain range. In future publications we will try to move existing time slots repetitively only a tiny bit (e.g. only several nanoseconds) over time. This should not interrupt the existing communication and enables drifting of existing time slots to reduce gaps. When this is done during reconfiguration larger gaps for inserted time slots should be achieved in the progress.

References

1. Bauernhansl, T.: Die Vierte Industrielle Revolution: Der Weg in ein wertschaffendes Produktionsparadigma [The Fourth Industrial Revolution - The Road to a Value-Creating Production Paradigm]. In: T. Bauernhansl, M. ten Hompel, B. Vogel-Heuser (eds.) *Industrie 4.0 in Produktion, Automatisierung und Logistik*, SpringerLink, pp. 5–35. Springer Vieweg (2014)
2. Burggräf, P., Dannapfel, M., Adlon, T., Riegauf, A., Müller, K., Fölling, C.: Agile Montage: Montageplanung und -system als integrale Bestandteile der Fabrikplanung [Agile assembly – Assembly planning and assembly system as integral elements of factory planning]. *wt Werkstattstechnik online* **109**(9-2019), 622–627 (2019)
3. Dürr, F., Nayak, N.G.: No-wait Packet Scheduling for IEEE Time-sensitive Networks (TSN). In: A. Plantec (ed.) *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, pp. 203–212. ACM (2016)
4. Hollender, M.: Collaborative process automation systems. ISA (2010)
5. IEEE: IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems (2008)
6. IEEE: IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks – Amendment 25: Enhancements for Scheduled Traffic 802.1 Qbv (2015)
7. Isaksson, A.J., Harjunoski, I., Sand, G.: The impact of digitalization on the future of control and operations. *Computers & Chemical Engineering* **114**, 122–129 (2018)
8. Kobzan, T., Schriegel, S., Althoff, S., Boschmann, A., Otto, J., Jasperneite, J.: Secure and Time-sensitive Communication for Remote Process Control and Monitoring. In: *Proceedings 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1105–1108. IEEE (2018)
9. Li, Z., Wan, H., Pang, Z., Chen, Q., Deng, Y., Zhao, X., Gao, Y., Song, X., Gu, M.: An Enhanced Reconfiguration for Deterministic Transmission in Time-Triggered Networks. *IEEE/ACM Transactions on Networking* **27**(3), 1124–1137 (2019)
10. Messenger, J.L.: Time-Sensitive Networking: An Introduction. *IEEE Communications Standards Magazine* **2**(2), 29–33 (2018)
11. Meudt, T., Pohl, M., Metternich, J.: Die Automatisierungspyramide - Ein Literaturüberblick: [The automation pyramid - An overview of the literature] (2017)
12. Nayak, N.G., Dürr, F., Rothermel, K.: Incremental Flow Scheduling and Routing in Time-Sensitive Software-Defined Networks. *IEEE Transactions on Industrial Informatics* **14**(5), 2066–2075 (2018)
13. Raagaard, M.L., Pop, P., Gutierrez, M., Steiner, W.: Runtime reconfiguration of time-sensitive networking (TSN) schedules for Fog Computing. In: I.F.W. Congress (ed.) *2017 IEEE Fog World Congress (FWC)*, pp. 1–6. IEEE (2017)
14. Randell, B.: A note on storage fragmentation and program segmentation. *Communications of the ACM* **12**(7), 365–369 (1969)
15. Verein Deutscher Ingenieure e.V.: *Cyber-Physical Systems: Chancen und Nutzen aus Sicht der Automation: [Cyber-Physical Systems: Opportunities and benefits from an automation perspective]* (05.2013)
16. Wiendahl, H.P., ElMaraghy, H.A., Nyhuis, P., Zäh, M.F., Wiendahl, H.H., Duffie, N., Brieke, M.: Changeable Manufacturing - Classification, Design and Operation. *CIRP Annals* **56**(2), 783–809 (2007)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.