Presented at the 13th International Conference on Informatics in Control, Automation and Robotics (ICINCO), July 2016, http://www.icinco.org/

Software-defined Networking for Real-time Ethernet

Jia Lei Du and Matthias Herlich
Salzburg Research, Jakob Haringer Straße 5/3, Salzburg, Austria
{jia.du, matthias.herlich}@salzburgresearch.at

Keywords: Software-defined Networking, Real-time Ethernet

Abstract: Real-time Ethernet is used in many industrial and embedded systems, but has so far mostly been statically configured. However, in the future these network configurations will be required to change dynamically, for example for highly flexible production lines or even software upgrades in modern cars that add new features which require changes to the in-vehicle network. Software-defined networking (SDN) is already increasingly used to dynamically configure non-real-time networks. In this paper we explore the idea of a software-defined real-time Ethernet. We analyze the features of current real-time Ethernet protocols, the applicability of SDN and give an overview of potential advantages of software-defined networking for real-time communication which can enable features not achievable using current solutions. In the future this development will likely lead to more flexible, efficient and robust real-time networks.

1 INTRODUCTION

Real-time Ethernet (RTE) allows the use of cost effective, widespread and high-bandwidth Ethernet technology in industrial environments like automation, process control and transportation where one key challenge is real-time communication, i.e. communication with guaranteed upper bounds for latency and latency variations (jitter). Various solutions like Ethernet Powerlink, VARAN, Profinet and TTEthernet have been developed to extend standard Ethernet with real-time capabilities.

Typical RTE deployments in the past have been configured once to run without re-configuration for years or even decades. However, in the future RTE networks will need to be more flexible due to a variety of reasons: To produce small lot sizes in a production environment efficiently, the underlying network must support quick reconfigurations to fulfill new requirements (Dürkop, Jasperneite & Fay, 2015). Or in-vehicle networks could be reconfigured through software updates for example when a new driver assistance feature needs a higher sample rate from a proximity sensor.

In non-real-time networks software-defined networking (SDN) is a technology that provides a great range of freedom to flexibly and centrally reconfigure the network on-demand. The basic idea of SDN is to control network flows through a centralized intelligent controller with "dumb" forwarding devices in the data plane of the network (McKeown et al., 2008). By monitoring network-wide state, the controller obtains an up-to-date view of the network and can dynamically adapt flows as necessary. The concept of SDN allows a wide range of traffic engineering, security and other applications. For example, flows can be dynamically rerouted based on load, failure or security scenarios to provide certain bandwidth or latency properties, fast failover mechanisms or security services. From an economic point of view, through standardization and centralization, SDN has the potential to simplify and reduce costs for network setup and operation.

In this paper we will describe our idea to apply software-defined networking in real-time Ethernet networks to benefit from SDN advantages while keeping the deterministic properties of RTE. In detail we propose replacing the switches/hubs of real-time Ethernet solutions with SDN-capable switches. Note that we do not consider replacing the real-time protocols themselves but to extend RTE protocols by providing additional features that the use of SDN controllers and switches make possible. For this purpose we first describe SDN in the next section. Then we describe features typical RTE solutions provide. Finally, we discuss the advantages and disadvantages of using SDN in an RTE network and give an approach how to validate these claims.

2 RELATED WORK

Gopalakrishnan (Gopalakrishnan, 2014) and Kalman (Kalman, 2014) both consider how SDN can be used in industrial communication networks. Gopalakrishnan provides a general list of SDN features and gives some examples how the advantages could be applied to an IEC 61850-based network, but only mentions real-time capabilities in passing. Kalman focuses on hardware abstractions and the ability to automatically configure networks using SDN. While both consider the advantages of SDN, they do not focus on the specific requirements and advantages SDN can bring to *real-time* networks, but on industrial communication networks in general.

(Dürkop, Jasperneite & Fay, 2015) provides a high-level concept for the automatic configuration of real-time Ethernet solutions. Our paper focuses on the communication aspect in more detail and proposes using SDN as an approach for network (re-)configurations. Furthermore, automatic (re-)configuration is only one of the advantages we describe in this paper that software-defined networking can bring to RTE networks.

3 SOFTWARE-DEFINED NETWORKING BACKGROUND

In most conventional communication networks, traffic flows are established based on forwarding rules that are locally determined using distributed algorithms. In contrast to this approach, traffic flows in software-defined networks (SDNs) are centrally configured by network applications via so-called controllers. This effectively decouples the control plane, which determines where traffic is sent, from the data plane, which forwards packets to their destinations. When a packet that matches a rule arrives at a network device, the associated actions are performed. Possible actions include the modification of packet headers and the dropping or forwarding of packets. Figure 1 illustrates the interaction between lower layer SDN forwarding devices, the SDN controller with its applications, and RTE devices.

One standard for the implementation of software defined networks is OpenFlow (Open Networking Foundation, 2015). The OpenFlow standard defines a communication protocol between network switches and one or more controllers. The ideas in this paper can be applied to all SDNs, but we will use OpenFlow as example when illustrating our ideas.

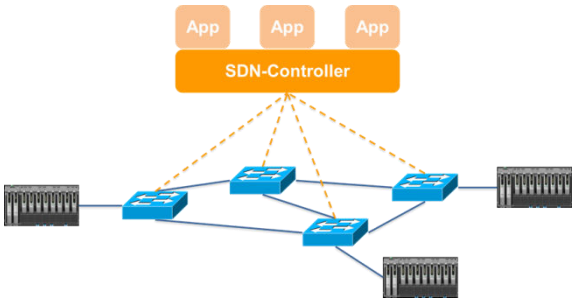

Figure 1. In SDN the network devices (middle) forward network flows programmed by a SDN controller (top) between end-devices (left/right). The SDN controller could also be integrated into one of the end-devices.

One key risk of an SDN is related to the availability of the controller that is required for configuring the network devices. Both the controller itself and the connection between network devices and controller represent possible single points of failures and bottlenecks. To mitigate risks of controller unavailability usually the use of multiple controllers in an SDN is suggested such as in (Yeganeh & Ganjali, 2012; Jain et al., 2013; Yazici, Sunay & Ercan, 2014).

4 REAL-TIME ETHERNET FEATURES

We have exemplarily chosen Ethernet Powerlink (Ethernet Powerlink Standardization Group, 2016), Profinet (International Electrotechnical Commission, 2014), TTEthernet (SAE Aerospace, 2011), VARAN (VARAN Bus User Organization, 2016) and TSN (Time-Sensitive Networking Task Group, 2016) (an upcoming but not yet finalized IEEE standard and the successor of AVB) for investigation. Industrial communication protocols like Ethernet/IP are implemented on application layer based on TCP/UDP over IP communication stacks. Protocols in this category are usually highly compatible and do not require special hardware or modifications. However, due to the use of the entire Internet stack cycle times are generally higher than those achieved by protocols implemented based on lower communication layers. The studied protocols are instead implemented directly on top of Ethernet and achieve significantly lower cycle times. Thus, the application of OpenFlow and software-defined networking to real-time networks of the second

category is technologically more challenging and findings and improvements are more likely to be transferable to protocols of the first category. Additionally, our selection of protocols covers both time triggered and polling-based protocols as well as protocols that use the entire Ethernet stack or only parts of the stack. For an analysis of the features of the studied real-time Ethernet protocols with respect to SDN we group the features in the following categories. (1) Performance: quantifiable measurements about the RTE solutions. (2) Compatibility: RTE solutions usage of standard Ethernet features. (3) Features relevant for SDN: Specifics of RTE protocols that are relevant for SDN.

The performance of an RTE can be described by cycle times and data rate. The cycle time is the duration of one transmission cycle, which is usually repeated as long as the network is operating. The cycle time is relevant for applications that need to transmit small amounts of data *often*. The data rate is the maximum achievable rate of data that can be transmitted over a single link under optimal circumstances. The data rate is important for applications that want to transmit large amounts of data. Both the cycles times and data rates given in Table 1 are for optimal conditions and are not necessarily achievable in practice.

RTE protocols are all based on Ethernet, but use different network modes and some change Ethernet standard formats. Network mode describes whether the RTE currently uses switches or hubs. All RTE protocols we consider can transmit non-real time traffic (for example web traffic) in time slots not reserved for higher priority traffic. VARAN uses its own kind of frame, while all other protocols we analyzed use standard Ethernet frames.

RTE solutions have two basic operating principles: Time scheduled and polling. In polling a single master server queries all clients according to its *internal* schedule. The clients are only allowed to transmit data in response to a query by the server. In a time scheduled network a *pre-defined* schedule is shared by all devices. The schedule describes which device is allowed to transmit at which time. While both time scheduled and polling based RTEs typically use a schedule, in polling the schedule is known only to the server and can be changed dynamically more easily. To use a distributed schedule precise time synchronization is necessary.

In case of link failures (such as cable breaks) some RTEs offer redundancy features, which automatically use alternate links to transmit the data and thereby hide the failure from the application. A broadcast (transmission from one-to-all devices) can be used to implement a multicast (one-to-some) by filtering out frames at the devices which are not intended to receive the frame. A more efficient method which we call real multicast is to transmit the frame only to the intended receivers in the first place. Using multipath routing several paths can deliver data from a source to a destination. This can be used for redundancy or to increase the data rate. We define concurrency as the ability of two pairs of senders and receivers to simultaneously communicate. This feature is, for example, easily achieved using switches, but not using hubs. Network topology describes the configuration of network devices the RTE solution supports. Hot plugging is the ability to connect and disconnect devices during network operation. Note that it is necessary to prepare the configuration for devices to be hot plugged in advance in some RTE protocols.

One of SDN's main capabilities is the fine-grained control of data flows in the network. Therefore, RTE features like broadcasting, real multicasting, concurrency, arbitrary topologies, redundancy and multipath routing will be as realizable using SDN as using more traditional networking approaches – SDN will potentially even allow a more efficient solution. However, one key limitation needs to be pointed out: Standard SDN devices currently do not support frame forwarding at precise points in time and, thus, do not naturally support time scheduled protocols. However, adding the notion of time does not conceptually contradict the use of SDN and is thus rather an implementation issue.

5 ADVANTAGES OF SOFTWARE-DEFINED REAL-TIME ETHERNET

In this section we discuss the advantages of applying software-defined networking to the design and implementation of real-time Ethernet. Some described features may already be supported or could be implemented with sufficient effort in existing solutions. However, even in those cases the use of SDN would still provide the advantage of being able to use an existing, consistent framework to implement all described advantages in a simpler way. Additionally, SDN enables features (e.g. the active use of network loops) that are not possible in existing solutions.

5.1 Advantages not related to path selection

Central configuration: Centralized software-based (re-)configuration of network devices is a key feature of SDN. It enables centrally controlled configuration of network nodes both with regard to device settings and communication patterns (this advantage has also been named "Flow Engineering" (Gopalakrishnan, 2014) or "Central Resource Management" (Kalman, 2014)). In difference to current RTE solutions where device settings and communication patterns are often configured once during design, using an SDN approach device settings and communication paths and schedules can be adapted on-the-fly with little or no disruption. From an application point of view a different production objective in a factory or a new feature in an autonomous vehicle could be activated through a software update even if the requirements towards the underlying RTE network changes, for example because certain sensor data is required at a higher rate or from a different set of connected sensors.

Standardization: First, OpenFlow defines a set of functionalities that all OF compatible network devices must fulfill and a standard interface to access these functions (also mentioned as "open standards-based and vendor-neutral" in (Kalman, 2014)). Second, as the intelligence is mostly located in the centralized controller, the network devices are comparatively simple. These two properties lead to simple, exchangeable, inexpensive, and future-proof network devices (except the SDN controller).

Global network information: OpenFlow-compatible network devices can collect a many usage statistics such as the number of received/sent frames/bytes per flow/port/queue. This information can help with error diagnostics and performance/traffic pattern evaluations. This feature is more valuable for real-time Ethernet networks in which the RTE controller does not already have a comprehensive overview of most or even all communication.

5.2 Advantages related to switching/routing/path selection

Central addition and removal of network nodes: Based on OpenFlow network nodes can be dynamically added to or removed from the real-time network at network level and removed nodes would no longer receive messages. Using this feature machines, sensors or actuators can, for example, be dynamically recombined to fulfill different tasks.

Arbitrary topology: Currently existing protocols usually support only standard Ethernet topologies and do not permit the existence of loops on network level and algorithms like spanning trees protocols are used to block redundant paths. Due to the central configuration of communication paths the existence of loops does not pose a problem for SDN and arbitrary network topologies can even be actively exploited.

Fast reroute and failover: Additional links in the network can be used as backup routes in case of failures in the network. This feature can be more

Table 1: A comparison of real-time Ethernet protocols and features and their relevance for SDN

	TSN	**Profinet**	**TTEthernet**	**Powerlink**	**VARAN**
Performance					
Min. cycle times	30 µs +	31.25 µs	<100 µs	<100 µs	<100 µs
Max. data rate	1 Gbit/s +	100 Mbit/s	1 Gbit/s	100 Mbit/s	100 Mbit/s
Compatibility					
Network devices	Switches	Switches	Switches	Hubs	Hubs
Non-RT traffic	Yes	Yes	Yes	Yes	Yes
Ethernet frames	Yes	Yes	Yes	Yes	No
SDN relevant					
Operating principle	Time schedule	Time schedule	Time schedule	Polling	Polling
Redundancy	Yes	Ring/multi-controller	Dual and triple	Ring and dual	No
Real multicast	Yes	No	?	No	No
Broadcast	Yes	Possible/not used	Yes	Yes	Master to slaves
Multipath routing	Yes	No	Yes	No	No
Concurrency	Yes	Yes	Yes	No	No
Topologies	Arbitrary	Line, tree, star, ring	Line, tree, star, ring	Line, tree, star, ring	Line, tree, star
Hot plugging	Yes	Yes	?	Yes	Yes

easily implemented for polling-based RTE protocols which often use broadcasts. In case of link failures, frames can be rerouted (Pfeiffenberger et al., 2015) transparently for end nodes as long as the frames arrive in time. For time-scheduled protocols the schedules in the network devices might have to be adapted after an incident to avoid congestion in the backup paths. For zero-loss/zero-time failover, flows can be duplicated on the network layer and delivered via two distinct paths.

Multiple simultaneous communication paths: Additionally available network links cannot only be used as backups in case of failures but also to increase available bandwidth during normal operation. Even multipath routing is imaginable, that is, splitting up and delivering flows via multiple paths.

Multiple networks over one infrastructure: An OpenFlow-/SDN-based approach to RTE networks could enable or simplify the operation of multiple real-time Ethernet networks over a single physical infrastructure, for example, in the most simple case by reserving half of the time for network 1 and half of the time for network 2. The devices in the two networks would never receive messages from the other network and thus this sharing of the physical infrastructure could be completely transparent to the participating devices. Such a setup may require some form of time synchronization between devices in the two networks which could take place in a third virtual network. This feature could be highly attractive for many polling-based protocols as such an operation can currently not be supported (due to the use of broadcasting for communication). For some time-based protocols like TTEthernet such a behavior could already be supported conceptually but the use of SDN would still significantly simplify the implementation by guaranteeing safety properties (e.g. nodes in network 1 will never see messages from nodes in network 2) similar to a virtualization layer in computing.

Isolation of faulty nodes: Using OpenFlow faulty network nodes can be easily disconnected from the network in the sense that messages of faulty nodes can be simply dropped at the closest functioning network node. The isolation of faulty network nodes consists of two separate problems: The detection of faulty behavior through the RTE and/or SDN controller and the disconnection of the faulty node through the SDN controller. Detection of very basic faults can, for example, be done through simple SDN-based frame counting. For the detection of complex faults the cooperation between RTE controller and SDN controller is likely necessary. Even a selective isolation of a node is possible: correct frames are allowed to pass and only incorrect frames that are sent at the wrong time or to wrong destinations are blocked.

Dynamic load balancing: Dynamic load-balancing allows the dynamic change of communication paths and/or the simultaneous use of multiple communication paths between a sender and a receiver as a function of network load. Within the scope of this paper/project we use the term only in the context of asynchronous traffic which potentially has more volatile communication patterns that are not known beforehand but less strict latency requirements compared to isochronous traffic.

Efficient multicasting: When delivering multicast traffic using OpenFlow, it is comparatively straightforward to avoid sending frames over a link if there is no subscriber of that multicast traffic at the other end of the link. In difference to standard Ethernet implementations where multicast frames are actually broadcasted in the network, this can both be a security benefit and to save bandwidth. More efficient bandwidth usage through efficient multicasting is possible for real-time Ethernet protocols which allow multiple parallel communication flows. And protocols that only allow one sender in the network at any time would still benefit from a security point of view as nodes that are not subscribers of the multicast traffic would not receive any of those frames.

6 DISADVANTAGES AND POTENTIAL PROBLEMS

Implementation of RTE using current SDN technology: The most important feature a RTE network has to implement is the deterministic guarantee of traffic latency. To make these guarantees usually polling or predefined communication schedules are used. If the creation of a polling-based software-defined RTE network was the goal, hubs would have to be replaced with switches. To implement a software-defined RTE network based on predefined schedules the SDN switches would additionally need to have a notion of time and schedules. While we do not know any conceptual reason which would prevent the support of time schedules in SDN switches, we are not aware of any standard SDN switches which support schedules. Additionally, when low cycle times are required, the performance guarantees depend on the achievable forwarding latency and jitter of SDN

switches. It is necessary to measure the performance of SDN switches and compare it to current Ethernet switches and hubs used for RTE. Finally, SDN in general does not dependent on the use of Ethernet-compatible frames, however current OpenFlow-compatible switches do pose that requirement.

Disadvantages introduced by SDN: One key disadvantage of SDN is the need for a controller. Such a controller is a single point of failure (if not replicated, see section II) and a controller failure would disable further central network configurations. However, this shortcoming prevents only use cases in which it is necessary to reconfigure the network while deterministic traffic is transferred over the network. In all other cases, the guaranteed performance would not be affected even if the SDN controller failed, only reconfiguration would be disabled.

7 VALIDATION CONCEPT

We are currently developing a proof-of-concept based on openPowerlink and SDN switches. openPowerlink is an open source implementation of the Powerlink real-time Ethernet protocol. A real-time Ethernet network with a cycle time of 1 ms has been built based on openPowerlink and OpenFlow-capable switches in our test lab. We are currently in the process of implementing key use cases to demonstrate some of the advantages described in this paper. Particular emphasis is put on demonstrating use cases which can be easily implemented using SDN but would be complex or impossible to implement using current standard RTE technologies. Finally, we focused on network level reconfigurations in this paper. However for the implementation of some of the described advantages, a tight integration and interaction with the respective RTE protocol would be necessary (e.g. to distribute new time schedules to the network devices). Thus, the long-term goal is to develop a complete software-defined real-time Ethernet solution in which the OpenFlow controller is integrated in the RTE devices and seamlessly interacts with the RTE protocols and its features.

8 CONCLUSION

We first described software-defined networking and features of real-time Ethernet solutions from a SDN point of view. Then we analyzed the advantages and disadvantages of the application of SDN approaches to RTE networks and described how we plan to demonstrate the advantages in practice. We conclude that the development of a software-defined real-time Ethernet is a highly promising endeavor and are in the process of validating our concepts in a test network.

(This work was partially funded by the Austrian Federal Ministry for Transport, Innovation and Technology in the project OpenheaRTEd, FFG No. 849972.)

REFERENCES

Dürkop, L., Jasperneite, J., Fay, A., 2015. An Analysis of Real-Time Ethernets With Regard to Their Automatic Configuration. In *IEEE World Conference on Factory Communication Systems (WFCS)*.

Ethernet Powerlink Standardization Group, 2016. Ethernet Powerlink Communication Profile Specification. Version 1.3.0.

Gopalakrishnan, A., 2014. Applications of Software Defined Networks in Industrial Automation.

International Electrotechnical Commission, 2014. Additional fieldbus profiles for real-time networks based on ISO/IEC 8802-3. IEC Standard 61784-2:2014, section CPF3.

Jain, S., Kumar, A., Mandal, S. et al., 2013. B4: Experience with a globally-deployed software defined WAN. *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 3-14.

Kalman, G., 2014. Applicability of Software Defined Networking in industrial Ethernet. In *IEEE Telecommunications Forum (TELFOR)*.

McKeown, N., Anderson, T., Balakrishnan, H. et al., 2008. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69-74.

Open Networking Foundation, 2015. OpenFlow Switch Specification Version 1.5.1.

Pfeiffenberger, T., Du, J. L., Bittencourt, P., et al., 2015. Reliable and Flexible Com. for Power Systems: Fault-tolerant Multicast with SDN/OpenFlow. In *7th IFIP Conf. on New Technologies, Mobility and Security*.

SAE Aerospace, 2011. Time-Triggered Ethernet. SAE Aerospace Standard AS 6802.

Time-Sensitive Networking Task Group, 2016. http://www.ieee802.org/1/ pages/tsn.html

VARAN Bus User Organization, 2016. “VARAN Real-Time Ethernet”.

Yazici, V., Sunay, M. O., Ercan, A. O., 2014. Controlling a software-defined network via distributed controllers. arXiv preprint, arXiv:1401.7651.

Yeganeh, S. H., Ganjali, Y., 2012. Kandoo: a framework for efficient and scalable offloading of control applications, *Workshop on Hot Topics in Software Defined Networks*.