DART: Dynamic Bandwidth Distribution Framework for Virtualized Software Defined Networks

Václav Struhár[1], Mohammad Ashjaei[1], Moris Behnam[1], Silviu S. Craciunas[2], and Alessandro V. Papadopoulos[1]
[1]Mälardalen University, Västerås, Sweden
[2]TTTech Computertechnik AG, Vienna, Austria

***Abstract*—In this paper we address a network architecture that uses a combination of network virtualization and software defined networking in order to reduce complexity of network management and at the same time support high quality of service. Within this network architecture, we propose a framework to be able to dynamically distribute the network bandwidth to various services such that the network resources are utilized efficiently. In many industrial domains, multiple services may use the same hardware platform for the sake of a better resource utilization. Therefore, bandwidth distribution among the services should be done in an efficient way during run-time. We also develop an admission control in this framework which dynamically coordinates the bandwidth distributions based on requested quality of services. We show the applicability of the proposed framework by implementing it on a common SDN controller. Moreover, we conduct a set of experiments to show the performance of the proposed framework.**

I. Introduction

With the advent of Industrial Internet of Things (IIoT), where manufacturing processes are monitored and supported by a tremendous number of devices, the need for flexible and efficient resource management in industrial networks is gaining its importance [1]. The emergence of IIoT brings intensification of resource sharing in physical networks, introduce new challenges in flexible network reconfiguration and challenges in providing various Quality of Service (QoS) levels in a physical network. To address these issues, both industry and research community are considering a combination of two emerging technologies, which are Network Virtualization (NV) [2] and Software Defined Networking (SDN) [3]. NV finds its roots in computing virtualization mechanisms where multiple virtual machines are running on a same hardware platform. Through NV a physical network is partitioned into several logical networks, known as *slices*, which are isolated and managed separately. Moreover, SDN on top of the NV provides an architecture in which the slices are managed via a centralized point without knowing the underlying physical network details. We use the term *virtualized SDN* for the described architecture throughout this paper. In this architecture, each network slice is managed by an SDN controller, and commonly the SDN controllers have different requirements when coordinating their associated slices.

This work was partially supported by the European Commission under the project FORA with grant number 764785.

Motivation. Commonly, in IIoT applications resources are limited, both in computation and in communication resources. Therefore, in the context of communication resources, several IIoT devices share a physical network to communicate. Although, the virtualized SDN architecture provides means in managing network resources, not much attention has been paid in supporting dynamicity of resource utilization which is a prominent factor in IIoT applications. Several works have proposed similar architectures focusing on QoS provisioning [4], network timing properties in real-time communication [5] and creating network slices according to application requests [6]. On the other hand, there are very few works addressing a fully dynamic network resource allocation in industrial systems. In this paper, we only focus on the network bandwidth as the resources could be energy or other constraints. For instance, the proposal in [7] attempts to develop an admission control in an SDN controller in order to bound the network bandwidth utilization for multiple network services. Nevertheless, the proposal is limited to a small network architecture without any ”smart” decision making algorithms for efficient bandwidth allocation.

Contributions. In this paper, we propose a framework, which we name it *dynamic bandwidth distribution (DART)*, based on a virtualized SDN architecture which makes the fully dynamic bandwidth allocation on a physical network feasible. Moreover, we propose an admission control mechanism to distribute the network bandwidth during run-time based on the QoS level requested by the IIoT devices. The admission control mechanism resides within the proposed framework. We also show the applicability of the proposed framework on a use case study where the proposed admission control mechanism is implemented within a well-known SDN controller. Finally, we conduct a set of experiments to present the performance of the implemented framework and mechanism.

Organization. The rest of the paper is organized as follows. Section II briefly describes the background and related work. Section III presents DART framework and the bandwidth distribution mechanism. Section IV presents the use case, while Section V shows a set of experiments. Finally, Section VI concludes the paper with future directions.

II. Background and Related Work

In this section we introduce the basic concepts of SDN and network virtualization as well as a survey in the area of dynamic bandwidth management.

A. Software Defined Network

SDN helps to decrease the complexity of network management by decoupling network control and forwarding functions [8]. Network control is handled in a centralized manner by an SDN controller that has a complete knowledge of the network. SDN is comprised of three layers (Fig. 1): a) The *Application layer* consists of SDN business applications written in common languages controlling the underlying SDN enabled devices via the SDN controller, b) The *Control layer* fetches various statistics from the physical devices (usage statistic, topology details, state details) and enables communication between SDN applications and SDN devices, and c) The *Infrastructure layer* is composed of physical SDN switches. The OpenFlow [9], [10] protocol enables communication between SDN controllers and network devices. The aim of the OpenFlow protocol is to overcome the proprietary systems of network hardware vendors and create a set of communication instructions for the interconnection of multiple vendors devices.

SDN switches contain hierarchically-chained flow tables defining rules and actions for handling incoming network traffic by SDN controllers. Flow tables are comprised of the following fields [10]:

- **Match fields**: The match fields consist of ingress ports, packet header fields, VLAN, priority and metadata.
- **Actions**: Actions to be performed for the matched data frame (e.g., forward data frame to a predefined port, drop the frame, send the frame to the controller). The actions can be chained in more complex actions.
- **Counter**: Statistic for matching data frames including count of data frames and their total sizes.
- **Priority**: Used if the incoming frame satisfies multiple match fields.

Every time a frame is received by an SDN enabled switch, the frame header is extracted and matched with records in local flow tables. If the match is found, the corresponding action is triggered. Otherwise, a copy of the frame is forwarded to the SDN controller that decides an action for the incoming frame. The action together with the matching rule is returned to the switch that stores in the flow table.

B. Network Virtualization

The need for service isolation and diverse resource requirements within one physical network brings the topic of network virtualization into the focus of the researcher community [11], [12]. Network virtualization enables the coexistence of multiple logical networks sharing the same underlying physical network [2]. One technique in this context is network slicing [13] where there is a division of the shared physical network into multiple logical isolated sub-networks (slices). Besides being isolated from each other, slices may be optimized for different purposes (e.g., high bandwidth HD video streaming, low latency video gaming) [13]. Slicing allows infrastructure providers to adapt the sharing of the underlying physical network to customer requirements while at the same time providing isolation of the network resources. Network virtualization requires a Network Hypervisor which creates an abstraction layer on the top of physical hardware and allows the creation of virtual networks.

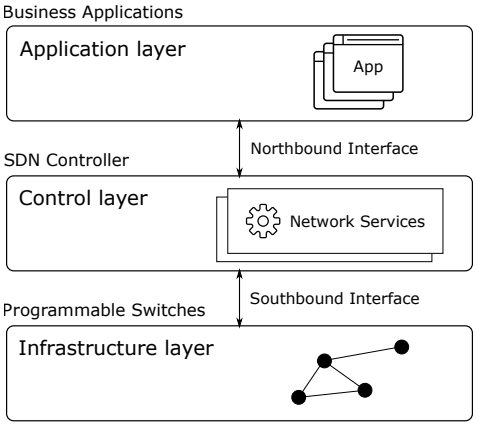


Fig. 1: SDN Architecture separates a network into tree layers: Application, Control and Infrastructure layer.

C. Bandwidth management

There are several resource reservation techniques in processor and communication domains which most of them focus on static reservation of bandwidth. For instance, in the processor domain, supporting multimedia applications [14] and hierarchical reservation techniques [15] are presented, whereas in the distributed level communication bandwidth reservation for multimedia systems [16], adaptive QoS control [17] and D-RES platform to support end-to-end timing [18] are presented.

In the context of using SDN architectures, Seokhong et al. [6] extended FlowVisor, as the network hypervisor, to guarantee the bandwidth requirements with an admission control and traffic scheduling. Moreover, Tomovic et al. [19] present a new SDN/OpenFlow control environment for dynamic adjustment based on Quality of Service (QoS) provisioning. In this solution, a centralized QoS SDN control system monitors the state of the network and automatically manages and configures network devices to provide the required QoS level for multimedia applications. There are several works, like HyperFlow [20], Onix [21], Kandoo [22], and devolved controllers [23], that use multiple SDN controllers on a single physical network (or a slice), either in a distributed or layered approach, where the orchestration goal between controllers is on controller redundancy and load balancing. The main focus is therefore to address the challenges that centralized SDN architectures introduce (c.f. [24]) rather than address network management. The closest work to this paper is a resource management solution for virtualized SDN networks in which each slice is governed by an SDN controller in terms of admission

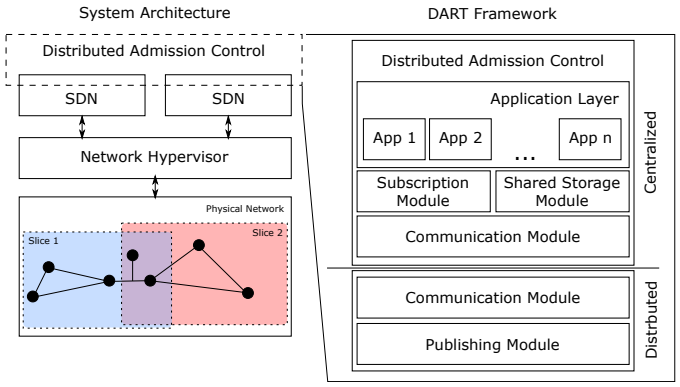


Fig. 2: An architecture of a system using DART framework.

control for incoming traffic [7]. However, the dynamicity was limited to static parameters and the SDN controllers are not communicating for a better decision on bandwidth allocation. Nevertheless, the mentioned works either focused on static bandwidth reservation or they proposed a level of dynamicity in bandwidth management with different goal than reducing resource utilization, which is the primary objective of DART.

III. DART: Dynamic Bandwidth Distribution Framework

This section presents the concept of DART framework as well as the proposed admission control mechanism. The framework defines an overall concept to enable dynamic bandwidth management of networks, while the admission control mechanism, as part of the framework, facilitates the functionality of dynamic bandwidth redistribution.

A. The DART Framework

The DART framework is a generic concept that can be applied on any virtualized SDN architecture. Fig. 2 depicts the DART framework on a virtualized SDN architecture. On the left side of the figure, an architecture with several slices in a physical network is shown. Note that in this architecture we assume that the slices can share a part of the pysichal network to increase the efficiency of utilizing the resources, however the framework covers the cases with fully isolated slices as well. The proposed framework is depicted on the right side of the architecture, which consists of two main components: a distributed component and a centralized component. Following we describe the components in details.

The distributed component deals with synchronizing the bandwidth management among the network slices. As each SDN controller can only coordinate its own slice, it is essential to have a general view of the network status when allocating the bandwidth. The distributed component ensures that the SDN controllers collaborate on bandwidth management, leading to a coherent bandwidth utilization. This component contains two main modules, which are communication and publishing modules. The communication module is responsible to send and receive information from and to the centralized component. The publishing module is the counterpart of the subscription module resided in the centralized component. Moreover, the coordination between SDN controllers are done using this component.

Another component in the framework is the centralized component which is responsible to coordinate the decision making for bandwidth allocation over the entire network. In case the changes in the bandwidth is requesting locally in one slice, the centralized component will decide on the allocation. However, if there is a shared part of the network that requires a change, SDN controllers need to negotiate on the bandwidth allocation from their slices. The centralized component is then responsible to advice the best possible bandwidth.

The admission control mechanism in this framework can be activated by any signal from various sources, including the load, priority and packet loss ratio, to start redistributing the network bandwidth. In this paper, as an initial phase, we specify priorities for the traffic to be sources of initiating the redistribution. Each IIoT device can transmit traffic with various priority levels depending on the QoS level it requires. Based on the traffic priority, the device can request for higher bandwidth during run-time. The details of this mechanism is presented in the following section.

B. Admission Control Mechanism

Following the DART framework, the admission control mechanism is divided into the centralized and distributed components. The centralized component contains the logic of the bandwidth management. The distributed component, however, resides on top of the SDN controller to provide information about the corresponding slices to the centralized component.

The sending nodes decompose the data into multiple data streams (see Fig. 3). The data streams can have different priorities which are set by the sender nodes depending on the importance of the data. Note that we consider only eight priority levels accepted by Ethernet frame. The primary goal of the admission control is to check the priority of the data streams and allocate bandwidth for the links that the data stream is transmitting. In order to do that the admission control defines a priority limit. Any received data stream with priority higher than the priority limit will be forwarded to its destination, whereas the data streams with priority less than the priority limit will be prevented for transmission. The priority limit is defined in the centralized component of the admission control for all slices. In a normal case, priority limits are equal and thus the bandwidth is uniformly distributed among the slices. If there is a request for priority limit change (detected by a distributed component), the centralized component adjusts the limits accordingly in order to a) increase bandwidth in the requesting slice, b) keep the total bandwidth used by all slices constant.

IV. Use Case: Surveillance System

To analyse the feasibility of the cooperation between the SDN controllers and to show the application of the DART framework, we present a use case of surveillance system in

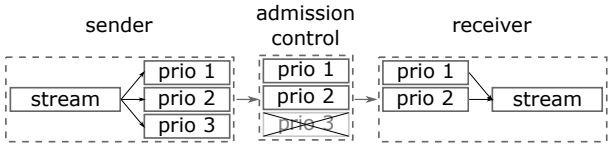


Fig. 3: Admission Control Mechanism.

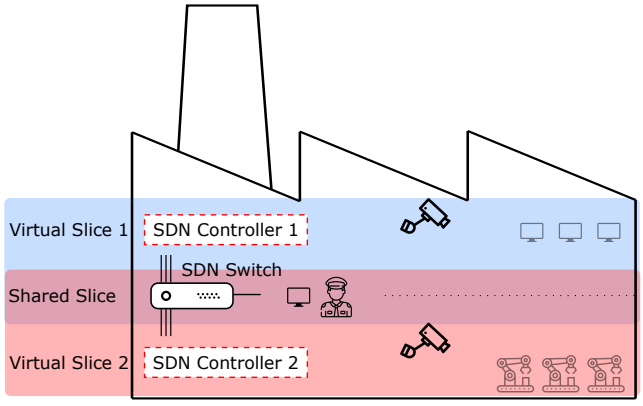


Fig. 4: The use case of a factory-wide surveillance system.

which the bandwidth is cooperatively adjusted based on traffic priorities. We present a factory-wide surveillance system (see Fig. 4), where the network is partitioned into several slices in order to keep virtual network domains separated (e.g., shop floor, administrative offices) and to allow to differentiate QoS levels across the slices. However, some of the services needed in both slices overlaps and must be available for both slices simultaneously. In the surveillance system, devices (senders) with attached cameras are transmitting video streams to a centralized location (receiver) where the video streams are processed and monitored by security guards in real-time. Senders have limited processing capacity that allows to perform simple motion detection algorithms. Based on the result they are able to ask for higher bandwidth by increasing priority of the traffic. SDN controllers monitor the corresponding virtual network slices and allow only traffic of certain priority levels to pass.

We have simplified the use case and implemented it as shown in Fig. 5. We consider three nodes. Two of them are sending nodes that are transmitting a video stream to the receiving node. The sending nodes are separated into two virtual network slices, these devices have cameras connected and periodically send sequences of images to the receiving node that stores and analyses the images. The receiving node resides in shared network slice and thus provides services for both of the slices.

The bandwidth in the shared slices is limited. Thus, there must be a traffic control in order to prevent network congestion in the shared slice.

A. System Setup

For the implementation, the following hardware and software components are used (see Fig. 5):

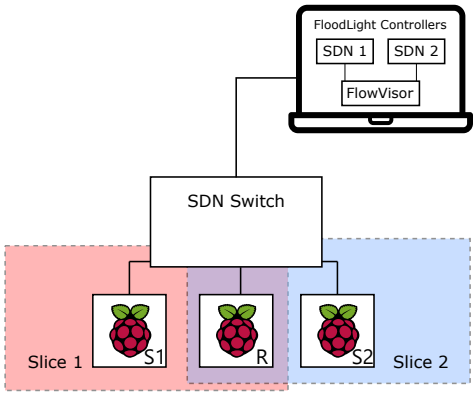


Fig. 5: The experimental setup.

- **Raspberry Pi (RPi) devices**: One RPi located on shared network slice acts as a data receiver, the rest of the RPi devices are separated on different network slices used as data senders. The data senders are connected to RPi cameras.
- **Aruba SDN enabled switch**: Three ports are used for connecting RPi devices, the fourth port is used for connection to the laptop that provides network hypervisor (FlowVisor) and SDN controllers (FloodLight).
- **FlowVisor**: FlowVisor is an OpenFlow controller serving as a transparent proxy between OpenFlow switches and SDN Controllers. It creates rich network slices defined by any combination of switch ports, src/dst ethernet address or type, src/dst IP address or type, and src/dst TCP/UDP port or ICMP code/type [25].
- **FloodLight**: FloodLight is an Java based open-source SDN controller offering a modular architecture that allows to extend its functionality with custom tailored applications.

B. System Implementation

The system requires implementation of four components: sending node, centralized bandwidth controller, SDN controller module and receiving node. With respect to DART, the bandwidth controller acts as a centralized component used for sharing states between the SDN controllers, and the SDN controller module represents the distributed part of the framework. The scheme of the implementation of the SDN controller module and the bandwidth controller is depicted in Figure 6.

The sending nodes are detecting motion in the video stream and transmitting video frames with corresponding priorities. The SDN controller modules are detecting priority changes in received data, reporting it to the centralized bandwidth controller, that decides the priority thresholds for corresponding SDN controllers in order to deliver higher importance video frames (with detected motion) with higher data rate to the receiving node.

a) Sending nodes: The sending nodes are continuously obtaining video frames, detecting movement, fragmenting the

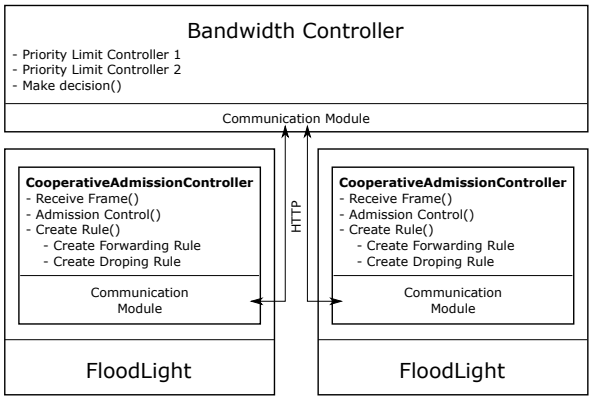


Fig. 6: Overview of the collaborative admission control using DART framework.

video frames into smaller segments and sending the segments to the receiving node at a constant rate. Based on the detected movement, the sender changes priorities of sent frames. The following steps are taken:

- Motion detection: The implemented motion detection algorithm compares a current video frame with the previous one, measures the amount of changed pixels and compares it with a pre-set threshold. If a motion is detected, the priorities in the sent data are increased.
- Video frame fragmentation: Due to payload limitation of IEEE 802.1Q Ethernet Frames (1500 Bytes), the video frame has to be fragmented into several frames. Payload of each frame contains a header (id of video frame, sequence of the the fragment, total) and video frame fragment data.
- Traffic differentiation: To differentiate the traffic to help SDN controllers to filter the data, the frames are uniformly distributed into data streams having priorities (0-5). If a video frame is detected, the sender increases the priority level of the traffic to (1-6).

b) SDN control: The FloodLight SDN controller is extended by a Java module that is governing incoming video frame streams. Based on priorities of the incoming data and limits imposed by the shared bandwidth controller, the data is either accepted and a forwarding rule to the shared slice is created, or the data transmission for particular priority level is blocked. The module performs the following:

- Receiving frames and creating rules: The module is receiving Ethernet frames from the SDN switch that needs actions to be resolved. Based on frame fields and custom data header, the the action the frame (MAC address of the sender, MAC address of the receiver, VLAN, priority) is established. There are two actions that the module can take: a) Create forwarding rule: Forward matching packets to receiving node, b) Create dropping rule: Drop matching packets.
- Collaborative admission control: The controller filters data frames by creating flow rules based with cooperation with the bandwidth controller.

c) Bandwidth controller: The bandwidth controller is implemented as a Java based application that provides functionality for a centralized bandwidth control, it provides REST API to enable communication between the bandwidth controller and the components distributed among SDN controllers. The application keeps track of the traffic in virtual networks and assigns priorities limits to distributed components residing on the top of the SDN controllers. The priority limits are assigned according to the Table I. The priority limits (S1 and S2 priority) are changed based on detected movement (S1 and S2 state).

d) Receiving node: The receiving node, located on the shared network slices, is collecting video frame fragments from the multiple slices, verifying data consistency and merging video frame parts in order to reconstruct the original images.

V. Experimental results

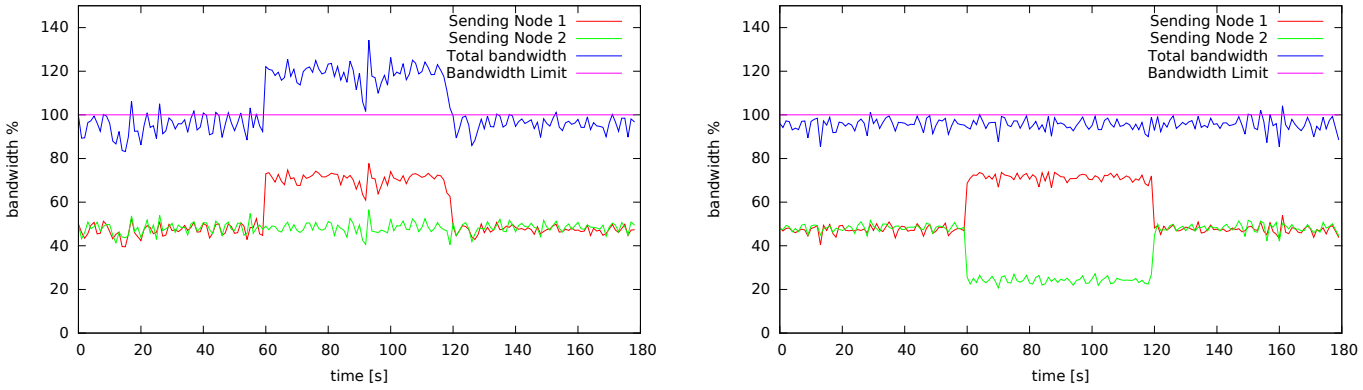
The purpose of the experiment is to show the behavior of the dynamic bandwidth allocation in sliced networks by a set of SDN controllers that are communicating through shared entity (the bandwidth controller). We consider transmissions from each slice with different importance that is changing dynamically during the experiment, in an event-based fashion. We assume that the traffic triggered by specific events, e.g., motion detection or alerts, are high priority traffic.

The sending nodes are transmitting sequence of video frames at a constant rate to the receiving node. The video frames have been prerecorded. The network hypervisor and two FloodLight controllers together with the shared bandwidth controller application run in a single computer, thus the network communication overhead between these entities is negligible. The node on a shared slice is receiving and reconstructing data. Also, the receiving node is recording all the arriving frames. The size and the sender of the frame is known, thus the network utilization can be reconstructed. The parameters of the system are the following: Image size ~ 40kB (depends on the scene, the image sizes may differ slightly).

Fig. 7 shows the average network utilization over 20 runs of the same experiment, in the case of no bandwidth adaptation (Fig. 7a), and with bandwidth adaptation (Fig. 7b). In both cases, between time 60 and time 120, a motion is detected, and additional traffic is generated from the sending node 1. In Fig. 7a, the bandwidth limit is exceeded for all the period when the motion is present, while Fig. 7b shows that the SDN controller 1 detects high importance traffic, and the system increases the priorities for slice 1 and decreases the priorities allowed for slice 2, resulting in a better allocation of the bandwidth over the high-priority traffic. The reaction time from the

TABLE I: Priority limits.

S1 state	S2 state	S1 priority	S2 priority
no movement	no movement	4	4
movement	no movement	6	2
no movement	movement	2	6
movement	movement	4	4



(a) Without bandwidth adaptation.

(b) With bandwidth adaptation.

Fig. 7: Average bandwidth utilization over 20 runs.

frame reception by the SDN controller, decision making by the bandwidth controller to the alteration of the rules in SDN switch is 60ms. The communication time between the SDN controller and bandwidth controller is 3ms.

VI. Conclusion and Future Work

Connection between network virtualization and Software Defined Networking plays an important role in Industrial Internet of Things due to the need for domain separation, provision of various levels of QoS in a single physical network and ability of dynamic network reconfiguration. Shared segments of virtualized networks can be used for sharing services and resources among separated segments. However, in order for efficient usage and utilization of shared resources, a collaborative approach must be set.

In this work, we introduced the DART Framework that enables collaboration multiple SDN controllers among virtualized networks. Subsequently, we implemented an use case of a surveillance system that utilizes the Framework. he results shows that SDN controllers can cooperatively take decisions and prioritize and distribute the bandwidth between slices to mitigate a congestion of shared resources. The Framework introduced here does not have to be restricted to bandwidth distribution only but it can be extended to support numerous application that will benefit from the inter-slice collaboration, e.g.: distributed access control lists, firewall and traffic scheduling.

The DART Framework opens the door for a design of smart algorithms for dynamic reconfiguration of the network that can utilize proactive monitoring of the flow/port usages in the separated virtual network.

References

[1] E. Sisinni *et al.*, "Industrial internet of things: Challenges, opportunities, and directions," *IEEE Trans. Ind. Inf.*, vol. 14, no. 11, 2018.

[2] N. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Computer Networks*, vol. 54, no. 5, pp. 862–876, 2010.

[3] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Comm. Mag.*, vol. 51, no. 2, 2013.

[4] M. Karakus and A. Durresi, "Quality of service (QoS) in software defined networking (SDN)," *Journal of Net. and Comp. Appl.*, vol. 80, no. C, 2017.

[5] K. Ahmed, J. O. Blech, M. A. Gregory, and H. Schmidt, "Software defined networking for communication and control of cyber-physical systems," in *ICPADS*, 2015.

[6] S. Min *et al.*, "Implementation of an OpenFlow network virtualization for multi-controller environment," in *ICACT*, 2012.

[7] S. Aglianò *et al.*, "Resource management and control in virtualized SDN networks," in *RTEST*, 2018.

[8] B. A. A. Nunes *et al.*, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Comm. Surv. Tut.*, vol. 16, no. 3, pp. 1617–1634, 2014.

[9] N. McKeown *et al.*, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.

[10] ONF, "OpenFlow switch specification," Open Networking Foundation, Tech. Rep., 2015. [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf

[11] L. Xingtao *et al.*, "Network virtualization by using software-defined networking controller based Docker," in *ITNEC*, 2016.

[12] A. Blenk *et al.*, "Pairing SDN with network virtualization: The network hypervisor placement problem," in *NFV-SDN*, 2015.

[13] R. Sherwood *et al.*, "Flowvisor: A network virtualization layer," *OpenFlow Switch Consortium, Tech. Rep*, 2009.

[14] X. A. Feng, "Towards real-time enabled microsoft windows," in *The 5th ACM International Conference on Embedded Software*, 2005.

[15] S. Saewong *et al.*, "Analysis of hierarchical fixed-priority scheduling," in *ECRTS*, 2002.

[16] M. Sojka *et al.*, "Modular software architecture for flexible reservation mechanisms on heterogeneous resources," *Journal of Sys. Arch.*, 2011.

[17] T. Cucinotta and L. Palopoli, "QoS control for pipelines of tasks using multiple resources," *IEEE Trans. Computers*, 2010.

[18] A. B. Oliveira, A. Azim, S. Fischmeister, R. Marau, and L. Almeida, "D-RES: Correct transitive distributed service sharing," in *IEEE Emerging Technology and Factory Automation*, 2014.

[19] S. Tomovic, N. Prasad, and I. Radusinovic, "SDN control framework for QoS provisioning," in *TELFOR*, 2014.

[20] A. Tootoonchian and Y. Ganjali, "Hyperflow: A distributed control plane for openflow," in *INM/WREN*, 2010.

[21] T. Koponen *et al.*, "Onix: A distributed control platform for large-scale production networks," in *OSDI*. USENIX Association, 2010.

[22] S. Hassas Yeganeh and Y. Ganjali, "Kandoo: A framework for efficient and scalable offloading of control applications," in *HotSDN*, 2012.

[23] A. S. . Tam, Kang Xi, and H. J. Chao, "Use of devolved controllers in data center networks," in *INFOCOM WKSHPS*, 2011.

[24] A. Hakiri *et al.*, "Software-defined networking: Challenges and research opportunities for future internet," *Comp. Net.*, vol. 75, pp. 453–471, 2014.

[25] "FlowVisor description," https://github.com/OPENNETWORKINGLAB/flowvisor/wiki, accessed: 2019-05-20.