In-Network Quality Control of IP Camera Streams

Csaba Györgyi, Károly Kecskeméti, Péter Vörös, Sándor Laki
ELTE Eötvös Loránd University
lakis@inf.elte.hu

Géza Szabó
Ericsson Research
Budapest, Hungary
geza.szabo@ericsson.com

ABSTRACT

Manufacturing processes are often monitored by IP cameras. The generated video streams can be transferred via a wireless link to be processed and used by remote industrial controllers that manage and configure the industrial task in real-time. However, the link has limited bandwidth and is not capable of transmitting the aggregated traffic of all the IP cameras. Moreover, the platform provider of the remote controller (e.g., cloud) might charge extra fees for high volumes of network traffic. To optimize the data transport, we propose an in-network video quality control method for IP camera streams that drops non-essential frames from temporally irrelevant IP camera streams even when bandwidth is available. Our solution introduces a high-level API through which the operator can define which camera streams are needed with high quality at which actuator positions/states of the industrial process. Our method assumes an aggregation point that monitors the actuators' states and reduces the quality of camera streams that are not important for the control process, selectively dropping a set of video frames. We have implemented a P4-based prototype of the aggregation point and show that the remote controller can be fed with high-quality video streams while meeting bandwidth limitations and potentially saving data transfer costs without modifying the end-points.

CCS CONCEPTS

• **Networks** → **Programmable networks**; **In-network processing**;

KEYWORDS

P4, Network Exposure, Video Quality Control, In-Network Computing

ACM Reference Format:
Csaba Györgyi, Károly Kecskeméti, Péter Vörös, Sándor Laki and Géza Szabó . 2023. In-Network Quality Control of IP Camera Streams. In *The Twenty-fourth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc '23), October 23–26, 2023, Washington, DC, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3565287.3617609

1 INTRODUCTION

Current factory environments utilize various types of sensors that continuously transmit their status. For example, temperature and light beam sensors forward their data in a structured form to industrial controllers running locally or at remote premises like edge or core cloud. These sensors are enough to transmit their status in every 10-ms or longer. However, an industrial environment is also equipped with IP cameras that can also be considered as special sensors generating larger amount of unstructured data. In contrast to traditional sensor streams, the transmission of their content at high frequency and high-resolution can significantly burden the communication network, particularly in the case of wireless networks. Furthermore, the platform provider of the remote controller (e.g., cloud) might charge extra fees for high volumes of network traffic.

Current operational technology (OT) in discrete manufacturing is focused on mass production with little options for product customization. The majority of wired networks used in factories are made up of a proprietary physical layer and other communication protocol technologies. These networks need time and money to build and maintain. Dealing with wires during production cell reconfiguration is a labor- and time-intensive task. This task can be made much easier with wireless technologies, which can also greatly speed up reconfiguration and open up new use cases. No wireless technology has yet been able to provide reliability and low latency that are on par with wired connections. With the arrival of 5G, this is changing, and operational technology companies are preparing to use 5G cellular networks to accomplish their Industry 4.0 vision [4].

Similar to how they do it now with wired network infrastructure, enterprises must connect the private 5G network, also known as the 5G non-public network (NPN), with their present OT/IT (Information Technology) systems, Industrial Internet of Things (IIoT) platforms, and control systems. The entire 5G system is intended to function as an essential component of the existing wired OT/IT communication infrastructure [14].

The IIoT exposure interface that 5G networks provide must be significantly easier to use than any of the current processes that rely on manual methods involving customer service requests sent to the communication service provider (CSP) or those that rely on CSP-oriented exposure interfaces that assumes in-depth familiarity with the internals of cellular systems. The interface must provide the proper level of abstraction so that factory operators may carry out their routine operational duties without the need for specialized assistance from the service and network provider [15].

Our method proposed in this paper builds on the assumption that not all the camera streams are always needed for the control during the operation of robots in the manufacturing area (e.g., production cell). The cameras used in the control process relies on the position and/or other internal states of actuators/sensors/robots.

Instead of modifying the IP cameras, we expose the quality control task to the network. A key benefit of this approach is the plug-and-play support of a wide range of cameras without special capabilities, and it does not require modification at the end-points. Assuming that the aggregated traffic of both camera streams and the actuators/robots is forwarded through a single aggregation point (e.g., a router connecting to a 5G network) to a remote industrial controller (e.g., running in the edge cloud). We reduce the quality and thus the data rate of selected camera streams that are currently not used in the control process. This selection is based on the current position (or other states) of the robot and high-level rules configured by the operator of the industrial task.

The proposed solution leverages programmable data planes running at the aggregation point that are able to observe both the state of the industrial setup and the video streams. Such a device is able to filter out packets belonging to a selected set of frames (e.g., P-frames in H264 [9] and H265 [18]) from the video stream without compromising the connection. This filtering is dynamically turned on and off — potentially also in the data plane — based on the observed state of the setup and the corresponding rules (e.g., feed of camera A is always filtered except when robot X is within a given area).

2 BACKGROUND AND MOTIVATION

2.1 Reconfigurable cameras

The traffic reduction can be realized with smart cameras providing configuration or computing APIs (Application Programming Interfaces) to alter their operation. However, such a feature requires special devices. Our contribution is complementary to this approach by not requiring any special functionality from the IP cameras and purely solving video quality control in the network.

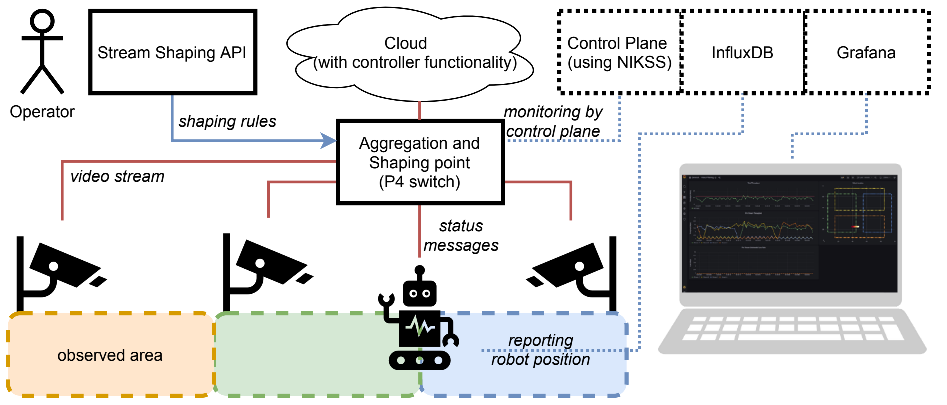
One such API set is provided by the Open Network Video Interface Forum (ONVIF). The ONVIF standard establishes a communication protocol for IP devices used in video surveillance and other physical security applications. The Media Service Specification [12] §5.5 sets up the Video Encoder configuration with parameters like resolution, quality, rate control. §5.5.5 "SetVideoEncoderConfiguration" operation needs to be triggered to modify a video encoder configuration. Note that client methods for changing a running stream are out of scope for this specification.

Note that the trade-off between application layer and in-network reconfiguration of the video QoS is that in-network reconfiguration influences the network immediately but may introduce issues on the video playback, while application layer reconfiguration provides no or less issues in video playback, but the reconfiguration takes time (at least seconds) to influence the network by any means.

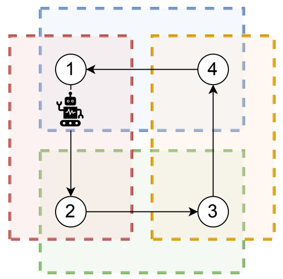
2.2 Network Exposure

The Service Enabler Architecture Layer for Verticals (SEAL) was introduced by Third Generation Partnership Project (3GPP) in Release 16 to enable factories and other verticals to automate system integration and 5G network configuration operations. The organization responsible for key communication applications and application enablers for vertical markets is the 3GPP Technical Specification Group (TSG) Service and System Aspects (SA) Working Group 6 (SA6). The purpose of SA6 is to offer 3GPP verticals with application layer architecture standards, including architectural requirements, functional architecture, processes, information flows, interoperability with non-3GPP application layer solutions, and deployment models where necessary. The APIs for provisioning, connection management, device management, connection monitoring, group management, user profile retrieval, identity and key management, location reporting, events, and network resource management are specified in 3GPP TS 23.434 [3].

Network Resource Management (NRM) in a radio network is crucial for efficient network operation, optimal resource allocation, enhanced quality of service, traffic optimization, spectrum efficiency, and network adaptability. It enables cellular operators to deliver reliable, high-performance services


Stream Shaping API
Operator
Cloud (with controller functionality)
Control Plane (using NIKSS)
InfluxDB
Grafana
shaping rules
monitoring by control plane
Aggregation and Shaping point (P4 switch)
video stream
status messages
observed area
reporting robot position

(a) The overview of the proposed solution. Italic labels denote the data communication. Dotted lines and components only serve observability/monitoring purposes.



(b) Observed areas and the path of the robot.

Figure 1: Overview of the proposed solution and an example scenario.

while effectively utilizing network resources and meeting the evolving needs of users. In traditional practices, NRM has been established and refined by the network operator. However, with the advent of programmable networks, developers now have a new tool that allows them to customize and configure NRM according to specific use cases.

3 SOLUTION OVERVIEW

The overview of our proposed solution is depicted in Fig. 1a. Robots in the manufacturing area are managed by an industrial controller running in the cloud. The remote controller relies on the streams of IP cameras deployed in the area. The robots also send status messages to inform the controller about their activity. Both the status messages and the video traffic flow through a programmable networking device (e.g., P4 switch). This serves as an aggregation point for the traffic of the manufacturing area. The network offers a quality control API through which high-level rules (e.g., a set of bounding boxes in which high quality video is needed for a given camera) can be defined for each camera stream and robot. These high-level rules are then mapped to low-level rules stored in match-action tables or compiled directly into the data plane. Based on the rules provided by the operator and the current status of the robots, the aggregation point can carefully drop packets from the video streams without compromising the required functionality. Note that the proposed method drops packets even when bandwidth is available.

For observability/monitoring purposes, we report the robot position and the network statistics to a time-series database (InfluxDB) that is visualized with Grafana or any other tool. Note that this is a one-way information flow that does not influence the behavior of the proposed solution.

4 DATA PLANE DESIGN

We used P4 [8] to implement the packet processing pipeline of the proposed solution. We distinguish two kinds of packets: *status messages* sent by the robots containing their position and *video packets* belonging to the video stream with H.265 payload over RTP. Video packets carry frames of the video. Since it is a real-time video stream, we have only two kinds of frames: I-frames that contain a complete picture at a given moment, and P-frames that reference and reuse parts of previous frames. Note that a single frame does not fit in a single packet, they are divided into so-called RTP fragmentation units. Based on certain fields, one can determine the type of the frame a given packet belongs to. Moreover, the first and last packets belonging to a frame are also indicated by one bit each.

Each video packet is mapped to a unique *stream_id* using a match-action table. Using this *stream_id* as an index, the current filtering mode is retrieved from the $filtering\_mode$ register array. For simplicity, we assume two filtering modes: *KEEP_ALL* to keep every packet, and *DROP_P* to drop the P-frames. If we are in *KEEP_ALL* mode, we do not drop any packets. However, if *DROP_P* is required, we are looking for the beginning of the next P-frame. Once we found it, we store a 1 value in the is_p_frame register array at index *stream_id*. Then, each packet belonging to the P-frame is dropped. Once we detect the end of the P-frame, the corresponding bit in is_p_frame is flipped back to 0.

Status messages carry the X and Y coordinates of the robot. In our case, they are simple UDP packets with two decimal values for the respective positions. Based on these values, the data plane updates the $filtering\_mode$ register array if required. E.g., when the robot enters the area observed by a specific camera, we require *KEEP_ALL* policy for the corresponding stream. Note that we do this entirely in the data plane, thus having lower reaction time.

Observability and testability is ensured by counters and meters provided by the P4 language as so-called externs. We use counters to keep track of the bytes sent and lost by individual video streams and in aggregate. The counter values are then accessible from the control plane. A meter is used to create an artificial bottleneck. Packets over the limit are flagged and later dropped leaving us space for updating our counters.

5 EVALUATION

Prototyping was carried out using a single IP camera (Reolink RLC-810A) and a cheap P4 programmable hardware platform (PCEngines APU4d4) [1, 10] using the PSA-eBPF P4 backend [13]. To scale up experiments, we built a testbed providing us with reproducible and more complex scenarios.

Our testbed is set up on a single server. We use the PSA-eBPF P4 compiler for the P4 switch component. The configuration contains four emulated cameras and a single robot. IP cameras are emulated by VLC instances streaming a video file. This video file is carefully re-encoded to match the frame patterns of a real camera having only P-frames with interleaving I-frames in every 2 seconds. The robot is emulated by a single python script reporting a predefined sequence of X and Y coordinates. The aggregation and shaping point is realized by a P4-switch using the PSA-eBPF backend. The emulated cameras and their receivers are placed in separate Linux networking namespaces and connected by the P4-switch using veth-pairs. The host's default namespace is also connected to the switch and used by the robot. To observe the traffic, the switch-state is constantly monitored by a script using NIKSS [13]. Values are inserted into an InfluxDB and visualized by Grafana. The robot positions are inserted directly into the InfluxDB by the script substituting the robot. The shaping rules are currently hard-coded in the P4 code, but run-time rule updates are also possible.

5.1 Scenarios

Using 3 different scenarios, we investigate the effect of bandwidth limitations on video streams and demonstrate our proposed solution that can remain well under this threshold. The results are also presented in our demo video [2].

We differentiate 2 kinds of loss. Packets that are lost because of the lack of available bandwidth are categorized as *unintended loss*. Packets dropped by our proposed method counts as *intentional loss*. For the sake of clarity we refer to the union of these two as *full loss*. A video stream can be in one of two different states. *Important streams* are crucial to the robot control functionality, thus they must be provided at the best possible quality (e.g., a robot is in the given area). *Unimportant streams* does not have quality requirements but must be kept alive (e.g, the observed area is inactive). We classify the streams based on the position of an emulated robot. The classification happens in the data plane based on the status messages set by the robot. The observed areas and the robot's circular movement is depicted on Fig. 1b.

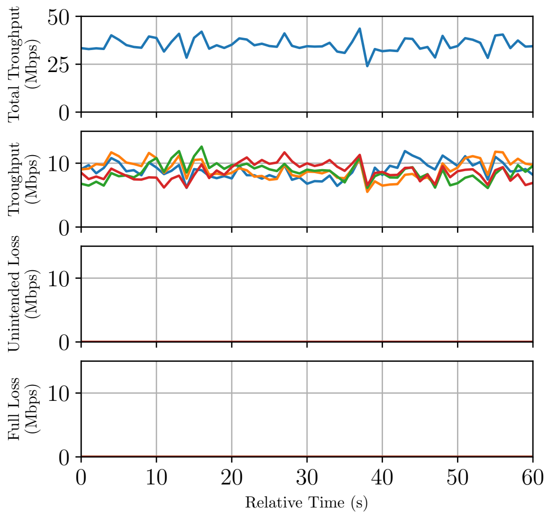
1) We leave the video streams unaltered without bandwidth limitation to observe the throughput needed to facilitate the four streams. The four streams generate around 30-40 Mbps of network traffic as shown in Fig. 2a. Since there is no bandwidth limitation, we do not observe any loss.

2) We introduce a bottleneck of 24 Mbps but leave the streams unaltered. Since this threshold is well below the required bandwidth, packet loss is inevitable. Packets are dropped evenly from every stream without considering their importance. Every loss we observe is unintended loss due to the lack of available bandwidth as shown in Fig. 2b. In this scenario, video streams suffer a significant quality degradation, rendering them indiscernible. This phenomenon is accompanied by significant packet loss that affect all the streams. Such streams cannot serve as basis for high-precision industrial control.

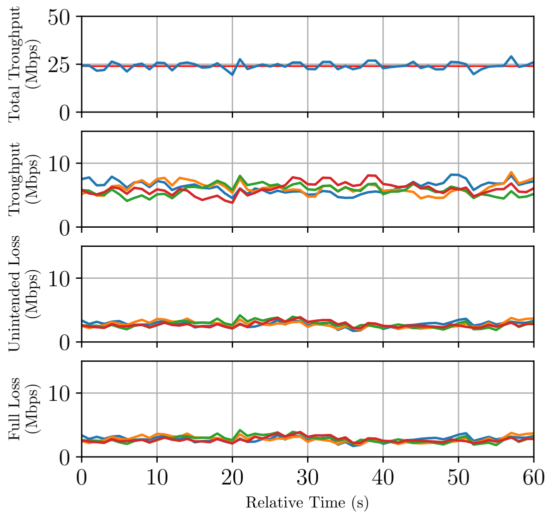
3) We apply our proposed solution. We keep the bottleneck from the previous scenario and drop any P-frames of streams deemed *unimportant* (based on the position of the emulated moving robot) leaving bandwidth for the *important* ones. Fig. 3 depicts the traffic patterns observed during a longer time interval with labeled with corresponding set-points of the robot. One can observe that streams classified *important* transmit at their original high throughput without encountering any loss. On the other hand, *unimportant* streams have a throughput close to zero and have high losses. However, these losses are *intentional losses* without causing any harm to the control functionality. *Unimportant* losses till have a non-zero throughput since I frames are still forwarded. One also observe the streams transitioning between states. Here we show that if we have two *important* streams at a time, the aggregated traffic fits well into the maximal 24 Mbps bottleneck capacity thus avoiding *unintended loss*. As a result, the *important* streams have perfect quality and due to the retained I-frames, we still deliver infrequently changing images for the *unimportant* streams, not breaking the connections of video streams.

5.2 Deployability

The proposed solution has a small memory footprint. It uses only 2 register arrays: $filtering\_mode$ and is_p_frame. Moreover, their size grows linearly with the number of video streams since each stream uses one slot in the registers.



(a) Multiple video streams sharing the same switch without bandwidth limitations.



(b) Multiple video streams sharing the same switch with 24 Mbps bandwidth limitation.

Figure 2: Introducing bandwidth limitations can lead to unintended packet loss in the video streams.

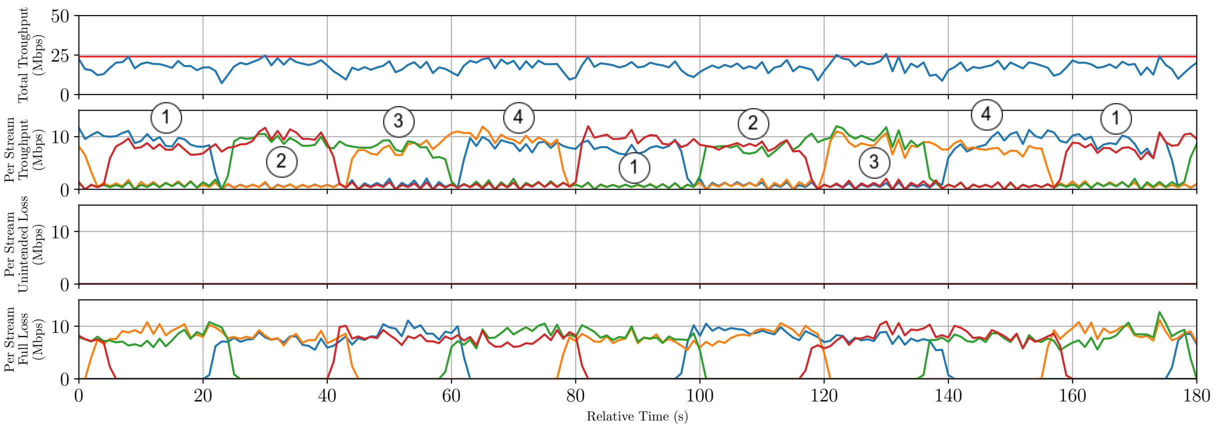


Figure 3: The proposed method in action.

Dropping P frames can be instantaneously started in the data plane once the corresponding *filtering_mode* slot is updated by a status message. Turning off the filtering is also instantaneous but the stream might only become distortion free at the next I frame. This can be mitigated by adjusting the time between I frames or by defining looser bounding boxes. Furthermore, using additional elements of the robot state makes a fine-grained on-off switching possible.

We used the flexible PSA-eBPF backend during our prototyping. Tough our experiences suggest that the data plane design can be modified to run on The Intel Tofino ASIC, the current use case does not require Tbps speed since a realistically high capacity up-link is about 1 Gbps.

6 RELATED WORK

In-network computing utilizing P4 allows the implementation of intelligent packet processing algorithms directly at the network edge. In addition to packet forwarding, programmable network devices can execute sophisticated tasks such as packet analysis, deep packet inspection, and content-based decision-making. This approach enables faster and more context-aware processing of network traffic, leading to improved network performance and responsiveness.

Researchers have proposed various techniques to leverage P4 for content-aware video streaming. For instance, a recent study by Wang et al. introduced a Media-Aware Network Element (MANE) [17] for intelligently streaming scalable video sequences in the P4 programming language. The MANE selectively drops queued scalable video packets when the queue occupancy exceeds a threshold, optimizing the streaming experience by efficiently managing network resources. Another approach proposed by Abiri et. al. is the combination of scalable video coding (SVC) and traffic offloading within a P4-based framework [5]. This scalable video traffic offloading (SVO) approach aims to enhance video streaming services in 5G heterogeneous networks by leveraging P4's programmability and traffic management capabilities. To address bandwidth competition among video clips, Ying et. al. propose the Content-Aware distortion-Fair (CAF) video delivery scheme [11]. This approach incorporates an understanding of video frame characteristics and guarantees a fair sharing of maximum-minimum distortion among video flows. CAF utilizes content awareness to prioritize packet dropping during congestion situations.

In addition to leveraging P4 and in-network computing for video streams, IP cameras can be "smartened" by integrating advanced capabilities such as AI-based video analytics directly into the camera itself. Ahmed et al. developed a prototype implementation of an AI-powered threat detection system called Hawk-Eye [6] for smart surveillance cameras. P4 was also shown to be useful in the cross-section of Active Queue Management (AQM) and video streaming. Almeida et al. proposed iRED [7], a P4-based implementation of the RED AQM, to improve the QoS of Dynamic Adaptive Streaming over HTTP (DASH). P4 has its uses in the realm of QoE as well; for example, Vogt et al. proposed QoEyes [16], an in-network QoE estimation of virtual reality technologies designed for the data plane.

7 CONCLUSION

In this paper, we proposed an in-network video quality control method for IP camera streams that drops video frames from nonessential streams even when bandwidth is available. We introduce a high-level API through which the operator can define which camera streams are required at which states of the industrial process. Our solution can decrease the load on wireless-, or other critical links with limited bandwidth. It can also lower traffic volume on the cloud side, saving traffic-based operational costs, as charging is often based on the amount of transmitted data.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Commission thought the HORIZON 6G SNS JU DESIRE6G (G.A. 101096466). The research was co-funded by the project Strengthening the EIT Digital KIC in Hungary (2021-1.2.1-EIT-KIC-2021-00006), implemented with the support provided by the Ministry of Innovation and Technology of Hungary from the National Research, Development and Innovation Fund, financed under the 2021-1.2.1-EIT-KIC funding scheme.

REFERENCES

[1] 2022. P4edge. https://p4edge.net/. (2022). Accessed: 2023-06-07.

[2] 2023. YouTube video of this demo. https://youtu.be/eXvYU3umr_c. (2023). Accessed: 2023-06-08.

[3] 3gppTS23434 2021. 3GPP TS 23.434, Service Enabler Architecture Layer for Verticals (SEAL); Functional architecture and information flows. (June 2021). https://www.3gpp.org/ftp/Specs/archive/23_series/23.434/23434-h20.zip

[4] Karaagac Abdulkadir et al. 2023. Managing 5G Non-Public Networks from Industrial Automation Systems. In *2023 IEEE 19th International Conference on Factory Communication Systems (WFCS).*

[5] Majid Abiri et al. 2022. Scalable video traffic offloading for streaming services in 5G HetNets. *Multimedia Tools and Applications* (2022), 1–23.

[6] Ahmed Abdelmoamen Ahmed et al. 2021. Hawk-eye: An ai-powered threat detector for intelligent surveillance cameras. *IEEE Access* 9 (2021), 63283–63293.

[7] Leandro Almeida et al. 2022. iRED: Improving the DASH QoS by dropping packets in programmable data planes. https://doi.org/10.23919/CNSM55787.2022.9964949

[8] Pat Bosshart et al. 2014. P4: Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review* 44, 3 (2014), 87–95.

[9] Randell Jesup et al. 2011. RTP Payload Format for H.264 Video. RFC 6184. (May 2011). https://doi.org/10.17487/RFC6184

[10] Sándor Laki et al. 2021. P4Pi: P4 on Raspberry Pi for networking education. *ACM SIGCOMM Computer Communication Review* 51, 3 (2021), 17–21.

[11] Ying Li et al. 2009. Content-aware distortion-fair video streaming in congested networks. *IEEE Transactions on Multimedia* 11, 6 (2009), 1182–1193.

[12] onvifmediaservicespec 2022. ONVIF Media Service Specification. (Dec 2022). https://www.onvif.org/specs/srv/media/ONVIF-Media-Service-Spec.pdf

[13] Tomasz Osiński et al. 2022. A Novel Programmable Software Datapath for Software-Defined Networking. In *Proceedings of the 18th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT '22).* Association for Computing Machinery, New York, NY, USA, 245–260. https://doi.org/10.1145/3555050.3569117

[14] Gergely Seres et al. 2022. Creating programmable 5G systems for the Industrial IoT. *Ericsson Technology Review* 2022, 10 (2022), 2–12. https://doi.org/10.23919/ETR.2022.9934828

[15] Géza Szabó et al. 2021. Assessment of the Efficiency of 5G Network Exposure for the Industrial Internet of Things. In *2021 IEEE Conference on Standards for Communications and Networking (CSCN).* 52–58. https://doi.org/10.1109/CSCN53733.2021.9686079

[16] Francisco Vogt et al. 2023. QoEyes: Towards Virtual Reality Streaming QoE Estimation Entirely in the Data Plane. In *2023 IEEE 9th International Conference on Network Softwarization (NetSoft).* 267–271. https://doi.org/10.1109/NetSoft57336.2023.10175463

[17] Guan-Ru Wang et al. 2018. Streaming scalable video sequences with media-aware network elements implemented in P4 programming language. In *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium.* IEEE, 1–2.

[18] Ye-Kui Wang et al. 2016. RTP Payload Format for High Efficiency Video Coding (HEVC). RFC 7798. (March 2016). https://doi.org/10.17487/RFC7798