Dynamic Content Adaptation for Timely Delivery of Critical Data under Network Congestion

Jia Lei Du, Stefan Linecker, Rodrigo Pommot
Advanced Networking Center, Salzburg Research
Salzburg, Austria
Email: {jia.du, stefan.linecker}@salzburgresearch.at

Reinhard Mayr
COPA-DATA GmbH
Salzburg, Austria
Email: reinhardm@copa-data.com

***Abstract*—The ever increasing efficiency in sensor-actuator electronics and data transmission technologies is enabling another phase in the digital transformation in the form of billions of connected Cyber-Physical Systems (CPS). As CPS are grounded in the real-world, (near) real-time communication will be essential for many use cases. And this communication will take place over networks that will typically not be under full control of the CPS. For this reason, a solution for robust and interoperable communication of critical data between CPS in a global network is essential. In this paper a general framework for dynamic and interoperable content adaptation based on underlying network conditions is presented. First validation results show that the framework can be used to dynamically scale the quality and amount of transmitted data and thus maintain timely delivery of critical data.**

Keywords–Dynamic Content Adaptation; Adaptive Streaming; Congestion Control; Real-time Communication; Cyber-physical Systems.

I. INTRODUCTION

A Cyber-Physical system (CPS) is a system that integrates computation with physical processes. Transcending traditional, standalone, embedded systems, a CPS is a network of interacting entities [1]. CPS is strongly related to another concept called the Internet of Things (IoT). IoT is a global infrastructure of networked everyday objects (”things”), connected through interoperable communication technologies [2].

The ongoing TriCePS project investigates barriers and possible solutions concerning such interoperable communication technologies. With a myriad of network-connected entities, there will also be an immense number of different communication connections with varying qualities of connectivity. Caused by the stochastic behaviour of the underlying wired or wireless (usually IP) networks and applications that compete for bandwidth, the experienced Quality of Service (QoS) can be volatile. The size of the experienced variations depends on the ratio between the application requirements and the granted network resources. Managing this interdependence successfully without relying on exclusive and over-provisioned communication infrastructures is a main challenge in comparison to existing systems. In particular when time-critical data has to be transmitted strategies that maximize the robustness of throughput for the most relevant pieces of information are needed.

A. Adaptation strategies

Generally, there is a variety of possible adaptation mechanisms for dealing with network congestion.

1) Network adaptation: One possibility would be to adapt the network environment, for example by using some form of prioritization based on Weighted Fair Queueing (WFQ) [3]. However, this typically requires support from and proper configuration of the underlying network and is usually not well supported over networks that are not under own control such as the public Internet.

2) Time adaptation: Self-limiting the data rate which increases the duration a data transfer takes may also help to ease overall network congestion. This approach can be useful for communication that is not time-sensitive. A background download of a software update might be a good example where mainly data integrity and not timing is important.

For this adaptation mechanism we have successfully experimented with the use of different congestion control algorithms for data flows with different priorities. The approach uses an aggressive congestion control algorithm for high-priority data flows (e.g., CUBIC [4]) and a conservative congestion control algorithm for low-priority ones (e.g., TCP Vegas [5] or LEDBAT [6]). Such an approach can be helpful if there is no control over the underlying network and some data has loose timing constraints.

3) Data adaptation: The focus of this paper will be on the third mechanism which dynamically adapts the content to be send to the underlying network situation. For demonstration purposes we have chosen the use case of transmitting a live video from a surveillance camera. As network conditions become worse, the camera can switch from high-definition video to low-quality video (as in HTTP adaptive streaming), then gradually reduces frame rates down to 1 frame per second, applies grayscale filters and in the most extreme cases only transmits textual representations of detected objects and their movements or just signals if there has been any movement in the observed area or not. The surveillance camera use case has mainly been chosen for its visual attractiveness and easy understandability. The same framework developed in this project could be similarly applied to other use cases in which timeliness of data is essential and the normal amount of data transmitted can be temporarily significantly reduced (for example by only sending alarms instead of all sensor values).

B. Network model

TriCePS assumes a very general network model. Two communicating nodes A and B interchange information via a communication channel. This communication channel can span multiple network devices. Every route can be thought of as having a variety of properties including current available

 ISBN: 978-1-61208-815-0

bandwidth, round-trip time and number of hops. Some of these properties might be mostly static (like the number of hops between two devices in a wired, local network), whilst others may change continuously (like the available bandwidth between two devices when other network flows are competing for resources).

Additionally, some further general assumptions are:

- There is no control over the network devices, features or configurations between two nodes.
- The route between two nodes can be used by potentially many other devices, too.
- There is no general means to control or modify network stack behaviour.
- It is possible to influence user space application behavior on nodes.

II. Related Work

A summary of various approaches for application layer congestion control is presented in [7]. The discussed approaches include message bundling (combining messages to the same recipient into a single message), the use of a message dispatcher which sends/receives messages on behalf of other systems, conditional messaging where (meta-)data or context information in messages is used to reduce the overall amount of data or messages exchanged, the use of persistent connections so that connections are not torn down after usage but instead kept open for reuse, piggybacking of data unrelated to the main exchange, self-throttling where nodes adapt the frequency of their messages based on their importance, data compression, and finally delta encoding where only differences to previous data is exchanged. The authors also give real-world usage examples for each of these approaches and analyse the approaches' effects on protocol efficiency, message sequencing, latency, performance and code complexity.

In general, with application level adaptation, the application layer is typically informed about current network metrics and adapts accordingly. This is not a novel approach; adaptive codecs have been around for quite a while. The Adaptive Multi-Rate (AMR) audio codec, which is widely used in GSM and UMTS, is a prominent example. With AMR, different coding schemes can be used depending on link quality measurements performed on the receiver side [8]. If conditions are good, AMR strives for speech quality (high speech bandwidth, low error protection). If conditions are bad, AMR strives for robustness (compromising on speech quality but boosting error correction and limiting bandwidth needs).

Another prominent example for existing application-level optimization used by many video stream providers is HTTP Adaptive Streaming (HAS). HAS estimates the current available bandwidth and adjusts the quality of the video stream accordingly. Segments of content (chunks that comprise short periods of video material) are encoded with different quality levels and sender/receiver choose segments according to current bandwidth estimates. The Motion Picture Expert Group (MPEG) proposed a standard called Dynamic Adaptive Streaming over HTTP (DASH) [9].

Minerva [10] is a solution for achieving video Quality-of-Experience (QoE) fairness based on TCP congestion control-like algorithms. In case of network congestion, Minerva video clients' bit rates converge towards QoE fairness while also ensuring fairness to other TCP flows on average. Compared to the use of other congestion control algorithms such as CUBIC or BBR (which do not optimize for QoE), a QoE increase of up to 32% could be achieved.

NADA (Network-Assisted Dynamic Adaptation) [11] also suggests adaptive real-time media applications that adapt their video target rate and thus their sending rate based on both Explicit Congestion Notifications (ECN) from network devices and implicit congestion signals (delay, packet loss).

In difference to the solutions above, TriCePS follows a more advanced approach by allowing the type of content being transported to change. A general framework and example implementation is presented that not only allows to adapt the sending rate but actually switches between very different types of application data to be transmitted ranging from video to high quality images to extracted features in form of text files.

III. Architecture

In this section, the architecture of the solution is presented. From a granularity point of view, all adaptation mechanisms are performed at flow-level (as opposed, e.g., to application-level or device-level). This is a comparatively non-invasive approach as each data flow can be treated separately and gives CPS users the means to prioritize individual data flows as necessary. Each application will also create and control its communication sockets (as opposed to, e.g., let an intermediary create and control all sockets and passing the data back and forth to the applications).

1) Components: Figure 1 shows two nodes, A and B, that represent communicating applications [12]. Both nodes A and B use the TriCePS software library. The library consists of several components.

The network monitoring module manages a modular set of measurement methods that continuously monitor the network flows between A and B. It then supplies a set of network metrics for both library-internal use and for use by the application business logic. The main network information required by TriCePS is whether there is congestion on the used network paths and the two main network-related metrics that are of importance for the purposes of TriCePS are latency and bandwidth. Changes in one or both of these two main metrics will serve as potential triggers for adaptations in the communication behavior of TriCePS-enabled CPS.

The pipeline, with its handlers, is an idea that we borrowed from Facebook's Wangle [13] project, which itself adapted this from Netty [14]. With pipelines, "the basic idea is to conceptualize a networked application as a series of handlers that sit in a pipeline between a socket and the application logic". Each handler has a specific duty. For example encryption, framing, compression, encoding, conversion, etc. The aim of this level of modularity is to keep the complexity associated with handling the different communication mechanisms of a wide range of entities manageable. The pipeline module glues together the individual handlers. Data flows from the business logic of node A through all handlers in the pipeline through the network to the opposite node, where all handlers are traversed in reverse order. It is noteworthy that Figure 1 shows one specific, simplified example of a pipeline setup. The two pipelines in the example hold the same handlers (h1, h2, h3,

 ISBN: 978-1-61208-815-0

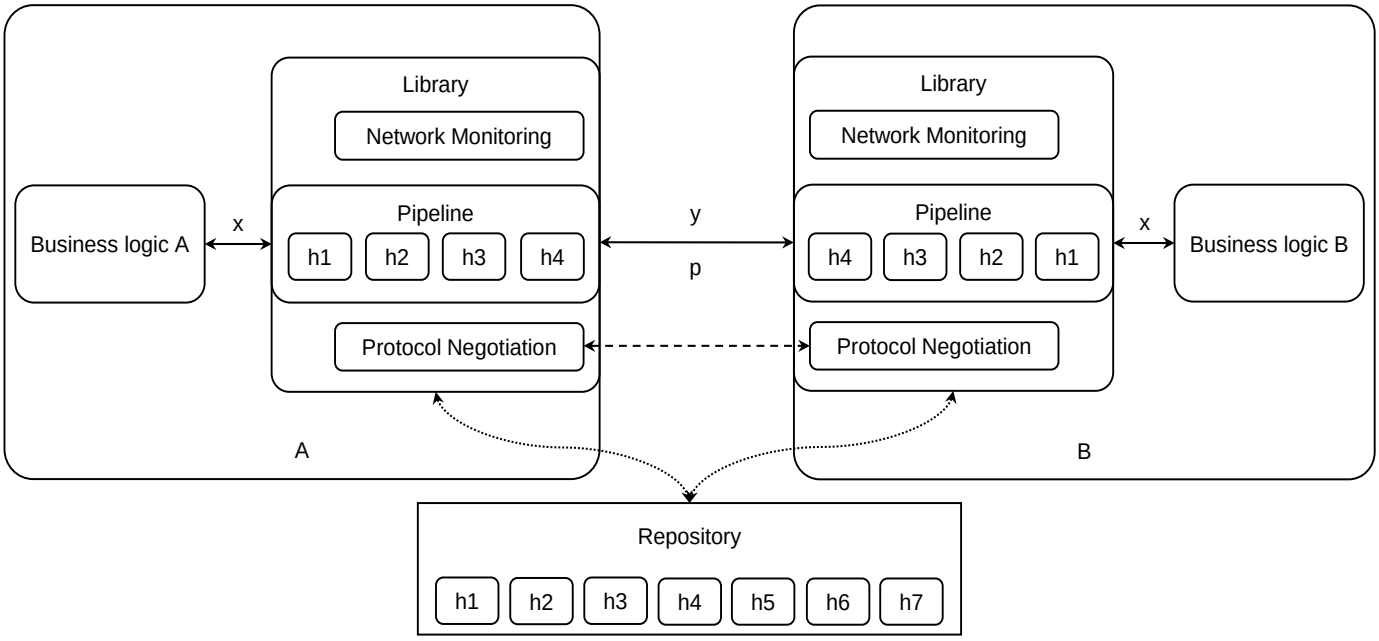

Figure 1. Overview of the reference implementation and its use by two nodes

h4) for each node. That is not a requirement, the two pipelines could each hold a different set and/or number of handlers if necessary.

Finally, the protocol negotiation module makes sure that the applications use pipeline setups that are compatible with each other. The mechanism has to be lightweight and fast to enable recurring, short-term (think seconds) pipeline reconfigurations due to network congestion. A repository acts as a library for new and updated handlers. The protocol negotiation module also takes care of the process of retrieving new or updated handlers from the repository.

It should be noted that the TriCePS concept assumes that all involved nodes use the TriCePS library and that the main constraint is limited network resources. The involved CPS need to have sufficient additional other resources (e.g, computing or memory) to execute the necessary TriCePS functions. As a workaround, the use gateway devices could also be considered.

2) In-band vs out-of-band communication of meta-data: Note that both the negotiation and the actual coordinated switching of handlers between TriCePS nodes require the exchange of metadata. This can happen in-band or out-of-band. Using an in-bad channel implies that the application stream has to be modified (e.g., negotiation packets inserted) and/or potentially even be (temporarily) paused for the negotiation/switching to occur. This can be comparatively intrusive from an application point of view. The advantage of using an in-band channel is that no second communication channel is needed as there might be scenarios in which the creation of such an additional channel may not be possible. An out-of-band negotiation/switching process can occur concurrently to the transmission of main data, avoiding any interruptions. Both approaches are supported by TriCePS, however, the latter is the preferred choice to allow for a more seamless operation.

3) Seamless Switching: To avoid any interruptions when performing a switch, the sender announces in advance after how many additional packets it will switch to a new pipeline. This switch is only performed upon confirmation from the receiver. Figure 2 shows an example of a switch after 4 additional packets. In this figure at first data is sent using pipeline “A” (black arrows). To start switching to pipeline “B” (blue arrows), the sender sends its current “send_counter” and the “switch_delay” (4 in this example) through the out-of-band channel. The sender keeps using pipeline “A” for 4 more times (black half-arrows) and only then it switches pipeline “B”. The receiver, upon receiving “send_counter” and “switch_delay”, will send a confirmation and then wait for its “receive_counter” to match the “send_counter”. Thus the receiver will process 4 more packages using the old pipeline and then switch to the new pipeline. This way a seamless transition can be achieved without any interruptions or modifications of the main data flow.

4) Stability and Fairness: In the case of multiple competing network flows, questions of stability (the desire to not switch between the modes too often) and fairness (on average every node gets about the same amount of resources) become relevant.

Two thresholds are defined for each send mode/pipeline: If the bandwidth (or frame rate) falls below the low threshold for some time, the mode changes to one that requires less network resources. If the bandwidth (or frame rate) rises above the upper threshold, the mode changes to one that requires more network resources.

To avoid that a node keeps switching between different modes too quickly and to avoid that multiple nodes switch to a higher sending rate and immediately back to a lower sending rate in a synchronized way, both a hysteresis and a probabilistic element are added to the switching component.

Upon hitting one of the two thresholds, a timer is started which serves as input to a sigmoid function which defines the probability p that a switch to a different mode is attempted.

 ISBN: 978-1-61208-815-0

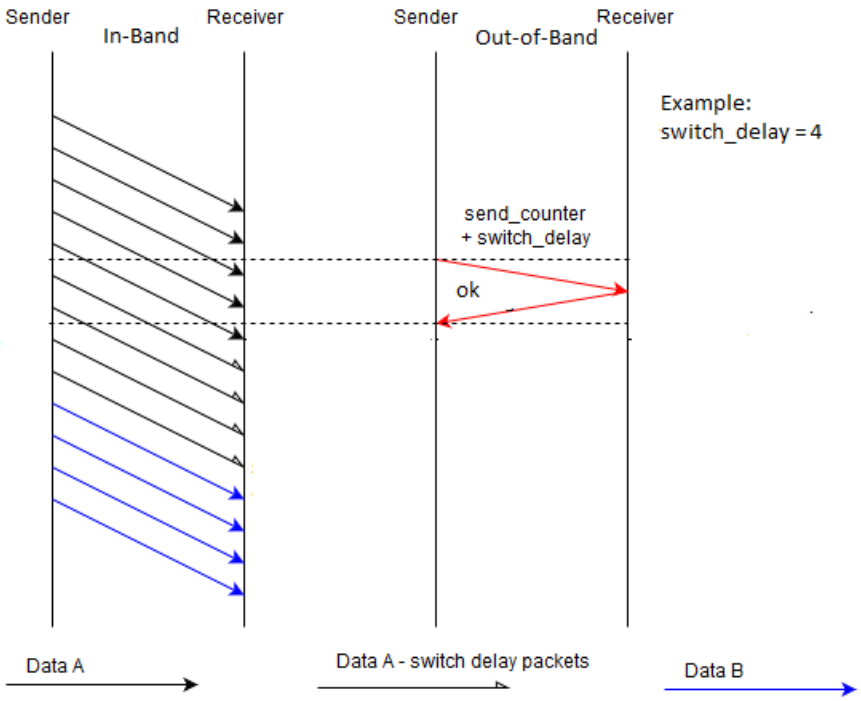

Figure 2. Implemented seamless switching mechanism

The timer is reset to 0 if the threshold criteria is not met anymore. The result is the longer a node remains below/above the lower/upper threshold, the higher the probability it attempts to switch to a different mode.

The specific thresholds and parameters for reducing or increasing the send rate can be chosen differently. In the example application, a down switch was triggered faster to avoid a reduced frame rate and an up switch was triggered more slowly as probing for a higher bandwidth too frequently can reduce overall system efficiency. This approach is quite generic and can be applied to any kind of transmission, however, the values for thresholds and the parameters may need to be adjusted for specific use cases.

IV. Implementation

A reference implementation of the TriCePS architecture was realized. We use a surveillance camera as example scenario and assume that the camera needs to transmit live digital footage over a communication network. The bandwidth requirement can be significant while the amount of available bandwidth fluctuates. Tackling this problem through (potentially extensive) buffering is not sufficient as the footage needs to be transmitted and processed in real-time.

To ensure liveliness of data, it can be better to compromise on video quality than to look at a stuttering and delayed high quality video. The amount of transmitted data is adopted according to the following hierarchy (from good to bad network quality):

- HD video, normal compression, normal frame rate
- Individual images at low frame rate, reduced quality
- Individual image features (using image feature extraction and sent through text descriptions)

This way relevant live information can be provided to the receiver even in case the available bandwidth is reduced by up to a factor of 100. The bandwidth requirements range from several Mbit/s for HD video to only several Kbit/s for textual descriptions of image features (see Table I).

Fig. 3 shows this mechanism as a state machine [12]. When network conditions are good, high quality video (as provided by [15]) will be emitted. When congestion is detected, lower quality still images will be emitted and when conditions get even worse, a text representation of the moving parts of the image will be emitted (the background image is only transmitted once and remains static). With improving conditions, the same process will happen in reverse order.

Note that the video data can be considered most valuable as this is the original data and the data reduction steps (conversion to single images, feature extraction) could still be performed at the receiving node if desired. The other way round, information that has already been discarded in the text representation cannot be reconstructed anymore (e.g., the arm and hand movements of the persons which have not been extracted).

The state where video is emitted uses a pipeline with two handlers (transcoding video to target bandwidth, framing), the state where still images are emitted uses a pipeline with three handlers (image extraction, image compression, framing) and the state where text is emitted uses a pipeline with four handlers (feature extraction, filtering of irrelevant features, encoding, framing).

1) Custom metrics: The network monitoring module manages a modular set of measurement methods that continuously monitor the network conditions between two TriCePS nodes (so, in this example case between the camera and the remote monitor). While the TriCePS software library provides the

 ISBN: 978-1-61208-815-0

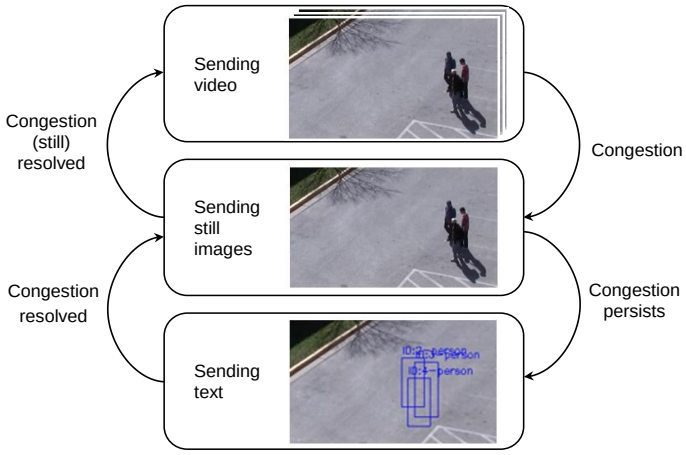

Figure 3. State machine for an adaptive camera

TABLE I. SENDING STATES AND THEIR PROPERTIES

State	*Updates/s*	*Bandwidth (kbit/s)*	*Bandwidth (%)*
Video	25	$\approx$ 2000	100
Images	1	$\approx$ 200	$\approx$ 10
Text	25	$\approx$ 20	$\approx$ 1

software interfaces to handle network measurements in a generalized way and already includes some standard measurements (e.g., round-trip delay or current sending rate), more application specific measurement methods can also be integrated. Since the camera use case is focused on image frames, we implemented a simple measurement method that measures the rate at which individual images can be emitted (frames per second).

This demonstrator serves as a visually attractive example application for the developed framework as it makes use of most TriCePS adaptations mechanisms and components at once. It will be subsequently adapted by industrial partner COPA-DATA for industrial use cases where for example near real-time availability of critical SCADA data is crucial.

V. TESTING AND VALIDATION

Figure 4 shows a basic test setup. A camera sends live images towards the display, and a traffic generator is used to create various degrees of network use/congestion.

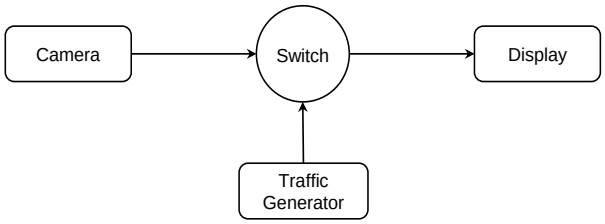

Figure 4. Schematic layout of the test setup

Figure 5 shows the measurement result of a test. Network congestion is generated roughly between seconds 30 to 60 and between seconds 100 to 160. The figure shows six curves, three of them depicting the frame rate in frames per second (fps_source, fps_control and fps_test) and the other three depicting the total frame count (frame_source, frame_control and frame_test). The curves with the ”source” suffix represent the count/frame rate of the camera. The curves with the ”test” suffix show the count/frame rate of the receiver in a TriCePS system. And finally, the curves with the ”control” suffix show what would happen without TriCePS. It can be seen that by prioritizing the timeliness of data (at the cost of quality), the frames/updates per second could be almost hold stable even in times of network congestion (fps_test recovers to stay close to fps_source after congestion sets in while fps_control remains low). Consequently, frame_test also follows frame_source much more closely than frame_control, showing a significantly smaller lag between between the camera output and the received data when using the TriCePS system. Overall, we observed significant improvements concerning number of successfully received frames/updates, delay of frames/updates and round-trip time (as a measure of network conditions) as summarized in Table II.

TABLE II. MEASUREMENTS WITHOUT AND WITH TRICEPS. MEAN VALUES AND STANDARD DEVIATIONS AVERAGED OVER FIVE RUNS

Value	*Without TriCePS*	*Using TriCePS*
Sent frames	2566 (72)	3995 (34)
Average delay	871 ms (146 ms)	108 ms (12 ms)
Average RTT	53 ms (4.5 ms)	33 ms (2.4 ms)

For larger-scale testing with up to 20 nodes, simulations with real TriCePS code in loop with a simple network simulation have been performed. When running multiple TriCePS nodes, on average flows get roughly an equal share of bandwidth (see Table III).

TABLE III. FAIRNESS MEASUREMENTS

# of flows	*Fair share of total BW*	*Min. share*	*Max. share*
2	50%	47.3%	52.6%
4	25%	22.3%	25%
8	12.5%	11.2%	13.4%
20	5%	4.16%	5.43%

VI. CONCLUSION

We have presented a framework for dynamic content adaptation for the timely delivery of critical data under network congestion. A pipeline-based data processing architecture with support for live negotiation and switching of pipelines forms the core of the solution. Using this approach a demo application that can scale the amount of sent data by orders of magnitude has been implemented. We have shown that average delay of critical data can be reduced by almost up to 90% (at the cost of data quality/amount). When multiple network nodes using the developed solution compete for resources, on average fairness of bandwidth allocation between the nodes is achieved.

ACKNOWLEDGMENT

This project is partially funded by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) under the program ”ICT of the Future” (https://iktderzukunft.at/en/).

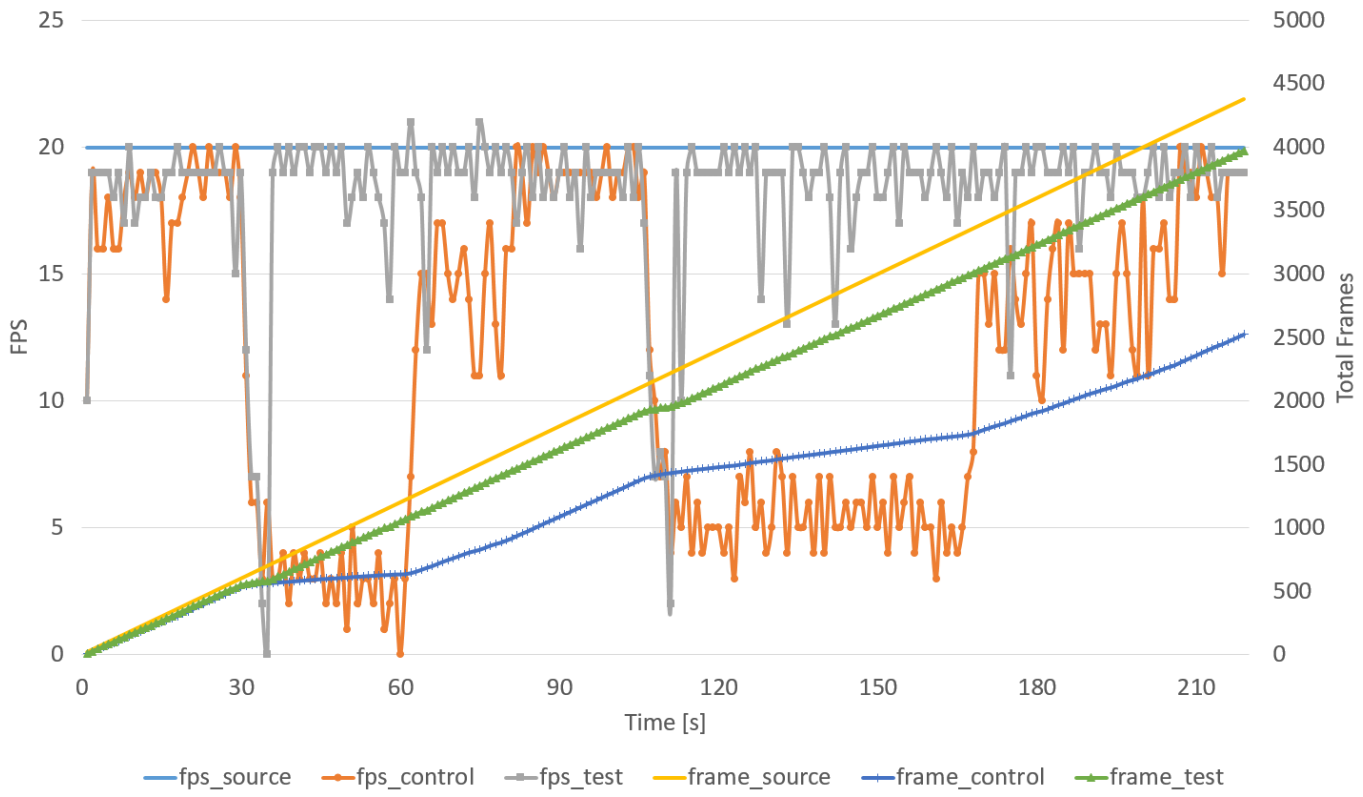

Figure 5. Frame rate and total frame count with and without TriCePS

REFERENCES

[1] R. Minerva, A. Biru, and D. Rotondi, "Towards a definition of the Internet of Things (IoT)," IEEE Internet Initiative, vol. 1, 2015, pp. 1–86.

[2] F. Xia, L. T. Yang, L. Wang, and A. Vinel, "Internet of Things," International Journal of Communication Systems, vol. 25, no. 9, 2012, pp. 1101–1102.

[3] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The single-node case," IEEE/ACM Transactions on Networking, vol. 1, no. 3, 1993.

[4] S. Ha, I. Rhee, and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," SIGOPS Oper. Syst. Rev., vol. 42, no. 5, 2008.

[5] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP vegas: New techniques for congestion detection and avoidance," SIGCOMM Comput. Commun. Rev., vol. 24, no. 4, 1994.

[6] S. Shalunov, G. Hazel, J. Iyengar, and M. Kühlewind, "Low Extra Delay Background Transport (LEDBAT)," RFC 6817, 2012. [Online]. Available: https://rfc-editor.org/rfc/rfc6817.txt

[7] P. Adamczyk, M. Hafiz, F. Balaguer, and C. L. Robinson, "Network congestion control at the application layer," in Proceedings of the 14th Conference on Pattern Languages of Programs, 2007, pp. 1–15.

[8] E. Ekudden, R. Hagen, I. Johansson, and J. Svedberg, "The adaptive multi-rate speech coder," in 1999 IEEE Workshop on Speech Coding Proceedings. Model, Coders, and Error Criteria (Cat. No. 99EX351). IEEE, 1999, pp. 117–119.

[9] I. Sodagar, "The MPEG-Dash standard for multimedia streaming over the internet," IEEE multimedia, vol. 18, no. 4, 2011, pp. 62–67.

[10] V. Nathan, V. Sivaraman, R. Addanki, M. Khani, P. Goyal, and M. Alizadeh, "End-to-end transport for video QoE fairness," in Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM '19), 2019.

[11] X. Zhu, R. Pan, M. Ramalho, S. de la Cruz, C. Ganzhorn, P. Jones, and S. D'Aronco, "NADA: A unified congestion control scheme for real-time media," Draft IETF, Mar, 2013.

[12] S. Linecker, J. L. Du, A. V. Silva, and R. Mayr, "The pipeline concept as key ingredient for modular, adaptive communication for cyber-physical systems," in Proceedings of the 17th International Conference on Informatics in Control, Automation and Robotics (ICINCO), 2020.

[13] Wangle Tutorial, (accessed January 20, 2020). [Online]. Available: https://github.com/facebook/wangle/blob/master/tutorial.md

[14] Netty: Home, (accessed January 20, 2020). [Online]. Available: https://netty.io/

[15] S. O. et al., "A large-scale benchmark dataset for event recognition in surveillance video," in Proceedings of IEEE Computer Vision and Pattern Recognition (CVPR), 2011.

 ISBN: 978-1-61208-815-0