

Controlling Intelligent Prostheses Using ETSI MEC

1st Gergely Gembela

Department of Automation and Applied Informatics
Faculty of Electrical Engineering and Informatics
Budapest University of Technology and Economics
Budapest, Hungary
0009-0004-7788-7903

2nd Gergely Mezei

Department of Automation and Applied Informatics
Faculty of Electrical Engineering and Informatics
Budapest University of Technology and Economics
Budapest, Hungary
0000-0001-9464-7128

Abstract—Currently, algorithms required to operate intelligent prostheses are usually executed locally, using the embedded system integrated into prostheses. However, the limited computational capabilities of such devices usually constrain the complexity and thus the accuracy of the algorithms that determine the intention of the user, resulting in a sub-optimal user experience.

Modern networking technologies in combination with edge cloud solutions offer significantly more raw computational power in comparison with embedded solutions. With predictable latency and low overhead, more robust and accurate algorithms could be applied for processing data collected by sensors, thus providing more accurate control, at the cost of the latency introduced by the network.

The goal of our research project is to determine whether 5G networks and edge cloud solutions are viable as the infrastructure for offloading parts of a control system to edge cloud applications and address the challenges of offloading control in time-critical settings with a relatively high-frequency input such as electromyography (EMG) sensors.

This paper describes a protocol we created for offloading near real-time computations using an internet protocol (IP) network, and presents the results of benchmarks conducted with a set of tools developed for benchmarking the capabilities of the protocol in different network conditions. The control of intelligent prostheses is an application the protocol is specifically tailored to. EMG data is collected and used to determine the intended action, with the accuracy depending on the complexity of a classifier algorithm. Other applications related to control systems with similar requirements were also kept in mind during the design phase. The capabilities of different networking technologies in combination with the framework are shown through the results of various benchmarks and experiments.

Index Terms—5g, MEC, Edge Cloud, Networking, Intelligent Prosthesis

I. INTRODUCTION

Intelligent prostheses are usually controlled with algorithms executed on an embedded controller integrated into the prostheses. With recent developments in wireless networking technologies and notable developments connected to edge cloud computing, offloading the processing of EMG data to the edge cloud could help minimize local energy consumption and provide a better user experience with better accuracy than previously possible. Multi-access Edge Computing (MEC) is a European Telecommunications Standards Institute (ETSI) Industry Specification Group (ISG) standard [1] that standardizes edge cloud computing, often deployed and used together with 5G and Beyond 5G (B5G) networks. The standard extends to other access networks such as Wireless LAN (WLAN) or Ethernet. MEC offers low latency and high bandwidth compared to traditional, centralized cloud services provided by Microsoft, Amazon, Oracle, IBM and more.

MEC is stated as the enabler of ultra-reliable, low-latency communications (URLLC), which could lift the solution described in this paper to the next level. However, when used together with regular and readily available networks and devices MEC already opens up plenty of new options without the guarantees of URLLC.

The 3rd Generation Partnership Project (3GPP) is working on a family of devices that implement only parts of the 5G standards referred to as RedCap¹ (Reduced Capability) devices. The technology is also referred to as NR-Light. The offloading of EMG processing fits the specifications listed for safety-related industrial wireless sensors and might provide a lower-cost alternative to URLLC solutions.

Unfortunately, as for now, most 5G deployments are non-standalone (NSA) as NSA 5G has no problem meeting the expectations of an average user (i.e. phone/mobile data users) and NSA is a natural step in the migration process from 4G to 5G. However, NSA networks lack key 5G functionalities, and the lack of 5G standalone (SA) networks slows the deployment of technologies like RedCap².

Intelligent prostheses usually measure EMG data, and based on the measured values a classifier algorithm determines the intended movement type and intensity of a given intent [2]. Although embedded the concept of System on a Chips (SoCs) in some cases featuring Artificial Intelligence (AI) accelerators is becoming more powerful and energy-efficient by the day, the gap between low-power solutions and edge cloud servers remains significant.

Embedded solutions usually use feature extraction before the classification of EMG data (e.g. [3]) which (in addition to complexity constraints) often results in less accurate but more efficient classification. In a MEC application, the use of a more complex classifier model or even an efficient combination of models is possible with an execution time comparable to that of embedded models. MEC applications also have enough resources to store measured data for further analysis. To achieve the best possible user experience, the ultimate goal is to minimize latency and maximize battery life and *accuracy*.

In the following chapters the paper outlines the goals and requirements we set, shows related work, then gives a brief overview of a protocol and a framework we created specifically for offloading the processing of EMG data. Finally, the results of measurements over various access networks are presented, and conclusions are drawn.

¹<https://www.3gpp.org/technologies/nr-redcap-glimpse>

²<https://www.fierce-network.com/broadband/lack-5g-standalone-networks-slows-redcap-deployment>

II. GOALS

Our goal was to determine whether current and/or next-gen wireless networking technologies in combination with edge cloud solutions can serve as a suitable alternative to on-device computations in compute-intensive control-related applications where the bandwidth requirement is moderate.

A protocol (along with a test framework), the subject of this paper was developed specifically for low-latency EMG data transmission, and an upper-bound to the maximum round-trip time (RTT) of 150 ms (including time dedicated for processing) was set, as above this limit the protocol could offer no improvement over the less accurate locally executed alternatives in terms of user experience. We determined if communications with latency below this limit are within reach, 5G (and other current- or next-gen wireless) networks might be capable of hosting an application that offloads the compute-intensive parts of a control loop.

It is worth noting that embedded SoCs are getting more powerful and energy efficient by the day while chip manufacturers seem to focus on *performance per watt*³. Thanks to industries such as AR/VR, embedded IoT and the TinyML⁴ paradigm companies invest more and more (funds and) resources into developing mobile/embedded technologies with dedicated hardware accelerators that make embedded AI solutions more efficient. In the future, the execution of more robust and energy-efficient models locally [4] might become a viable alternative to network-based offloading for some workloads.

While the protocol and the server-side classifier might serve only as a transient solution for controlling intelligent prostheses, the findings, measurements, the protocol and the offloading paradigm might remain relevant in use cases that rely on large or frequently changing server-side databases or models. The experience we gained by using 5G in real-time and/or control-oriented applications applies to a wide range of other challenges. Another major advantage of MEC-based solutions could be the ease of (potentially dynamic or even use-case-based) reconfiguration of the evaluation logic/algorithm.

III. REQUIREMENTS

In the following subsections, requirements are outlined that served as a base while developing the application layer network protocol that facilitates offloading of control to a MEC environment. Although the requirements originally derive from the main use case (EMG processing), they might also be applicable to use cases with similar networking requirements.

- **Minimal overhead, up-to-date state:** The nature of the task requires the protocol to deliver data in both directions as quickly as possible, with as little overhead as possible. The requirements (and expectations) are different for communication that controls the protocol itself and that transmits data, therefore two kinds of messages are to be distinguished and might be treated differently.
- **Extensibility:** The protocol should be designed to be easily extensible with new message types, tests, benchmarks and server-side modules that handle the received

messages. To leverage compiler optimizations and keep implementations as simple as possible, native dynamic extensibility is not required.

- **Security:** Communications over the protocol must be protected against malicious manipulation and interception.
- **Reliability:** In many use cases, reliability is a key concern. The protocol must be able to detect connection errors and needs to make the use of a locally executed fallback classifier possible. Note that we do not (and cannot) expect the same level of efficiency when processing the data locally, but this is hardly an issue, as the chances of requiring the use of the fallback classifier should be minimal under normal circumstances.
- **Configurability:** The protocol aims to support different kinds of EMG feature vectors with a configurable number of electrodes, ADC resolution, and window size (the number of vectors sent as a single message).

IV. RELATED WORK

Several surveys are available focusing on the performance, dependability and possibilities of MEC [5] [6].

Research of edge computing and offloading computations to the cloud or edge cloud environments (at least for now) is more focused on making the decisions whether to offload a task or not, and the analysis of those decisions (mainly focusing on cost both in terms of latency, and resources). Offloading control-related computations to the edge cloud fits the "Edge Native Design" paradigm outlined in [7].

Augmented/virtual reality (AR/VR) and computer vision-related applications are a prime use case of edge computing as both its latency and bandwidth requirements push networks to the limits, and rule out the use of traditional cloud computing. Research focuses mostly on bandwidth/throughput and QoS concerns [8].

Industrial IoT applications are often combined with private 5G networks [9] and solutions are focused on a generalised architecture rather than minimal overhead [10].

The integration of edge computing and machine learning is referred to as Edge intelligence [11], we found this to be the field of research closest to ours. Reviews [12] and [13] provide a detailed overview of the current state of EMG processing for use with arm prostheses. The latter identifies the lack of robust portable embedded systems as one of the major challenges.

Before designing a new protocol, a natural step was to go over existing protocols that could be applied to EMG data transmission. Protocols targeting embedded systems such as MQTT⁵ (along with MQTT SN and MQTT/UDP) and CoAP [14] were examined as alternative application layer protocols, but we determined that even though solutions exist both for IoT communications (e.g. CoAP and MQTT) and real-time streaming (e.g. RTP) none of these target low-bandwidth and low-latency use cases such as the transmission of EMG data, and a protocol specifically designed for this purpose could achieve lower overhead both in terms of additional data transmitted, and computational requirements.

V. THE *NWEmg* PROTOCOL

NWEmg is a protocol designed to make offloading EMG processing from intelligent prostheses to the edge cloud

³<https://newsroom.arm.com/blog/performance-per-watt>

⁴<https://newsroom.arm.com/blog/tinyml>

⁵<https://mqtt.org/mqtt-specification/>

simple, efficient and reliable. The protocol was designed to achieve the best possible performance.

While the guarantees provided by the URLLC 5G communications would make the protocol seem extremely robust, the protocol does not require the use of URLLC, and was designed to operate in all common modern wireless networking environments (5G/Wi-Fi6), and a fallback mechanism is expected to be available locally. Most classifier algorithms are stateless, therefore packet (and as a direct result message) loss is tolerable to a certain extent. The potential out-of-order reception of messages should be accounted for by the implementation.

Our protocol exchanges binary data with messages of predefined formats. A detailed specification is available on GitHub.

The protocol uses UDP as it needs to avoid extra latency potentially introduced by head-of-line blocking [15] and achieve the lowest possible overhead. The application layer protocol can be kept rather simple as the exact structure/size of datagrams is always known before transmission/reception.

The drawbacks of using UDP are that the protocol needs to handle connections, ensure the detection of reordering, packet loss or multiple reception and guarantee the consistency of the messages (as in IPv4, the UDP checksum is optional, and the 16-bit checksum is rather weak).

As per UDP Usage Guidelines [16] Datagram Transport Layer Security (DTLS) is used to secure communications over the protocol.

A. Messages

A message is the unit transmitted by the protocol. Over IPv4 usually 1 IP packet, over IPv6 strictly one IP packet. The maximum message size is determined by the configured maximal UDP datagram size. The default maximal datagram size the protocol supports is 1500 bytes, as path maximum transmission units (MTUs) are typically smaller.

The protocol has two different kinds of messages, control messages, and data messages. Control messages can be further divided into three categories based on the possible sender: (i) client (ii) server (iii) both parties. All messages (regardless of their type) are sent through the same UDP sockets, but depending on their type the way a given message is handled might be entirely different. The implementation should be capable of processing multiple messages asynchronously, regardless of the type of incoming messages. To help implementations reduce or even eliminate the need for dynamic allocations, all messages are either of a fixed length or have their maximum sizes specified. The only message type that has a non-fixed size is the *Data* message, but a safe upper limit to its length always exists.

Control messages carry information required to operate the protocol itself, ensure that the client and server-side state is synchronized, and take care of error handling. Control messages are prone to packet loss, thus control messages should be resent by the client in case the reply to the request times out. Only the client should send messages repeatedly, control messages have idempotent effects on the server's state, therefore with each client-initiated retransmission results in a new reply being sent to the client.

Client-initiated control messages are the *InitRequest*, *DisconnectRequest*, and all state transfer requests (i.e. *StartTrainingRequest*, *StartInferenceRequest*). These messages must be

sent from a client, when a client receives a client-initiated message, the result should be an *Invalid Message* error. Server-initiated control messages are *InitResponse* (the response sent to an *InitRequest*) and *StateReport*. When a server receives a server-initiated message, the result should be an *Invalid Message* error.

ErrorReport messages carry information regarding errors that may or may not be fatal to the session. Error reports should be of the following kind: i) Invalid message ii) Critical error iii) Invalid state transfer iv) Mismatched version v) Custom error. Error reports allow a variable length (max. 1024 byte long) message to be sent detailing the error. The length field holds the length of the message, the message field holds the UTF-8 encoded bytes of the error message.

StateReport messages used for state synchronization. The client follows the server-side state upon these messages. The message contains the state in which the state transfer was initiated (from) and the state after the state change (to), but state change messages with identical *from* and *to* states should be used to indicate when a state change request did not affect the server-side state. State transfer messages are not necessarily the result of *StateChangeRequests*.

Data messages handle the transmission of EMG data and classification results. The message contains the elements of the data vector packed without padding with LE byte ordering. Requirements for delivering data messages are unique, with the most important aspect being the up-to-date delivery of EMG data. These messages are never retransmitted. (*DataGuaranteed* messages (that carry data semi-reliably, but are an optional element of implementations) should be used when the loss of messages can not be tolerated. *DataResponse* messages are the responses to *Data* and *DataGuaranteed* messages, they carry the information required to operate the prostheses.

Under normal circumstances, it is not possible for the protocol to send or receive a message of type *unknown*, however, it seemed logical to create a message type that represents all messages received with an unknown message type. The reception of such messages should be treated as an error. Reasons for a server or a client receiving unknown messages could be undetected data corruption (though the chances of that happening are extremely low provided the 32-bit checksum), human error, or mismatched protocol versions (in case a new message type is introduced on one side, but not the other), but this case is accounted for during initialisation, as a compatibility check occurs.

B. Sessions

Even though UDP is connectionless, the protocol uses sessions. Here, sessions are temporary, stateful series of communications between the client and the server. Sessions are identified by 128-bit UUIDs, but sessions are directly associated with the IP addresses and ports of clients. The possible states and legal state transfers are shown with arrows in Fig. 1. Solid arrows represent regular state transfers, state transfers that are legal but occur as a result of a resolvable error (or an error recovery process) are shown with dashed arrows. State transfers are idempotent and illegal state transfers must always result in an error reply.

Both the server and the client are initialised with the state *Created*. This state indicates that the processes were started,

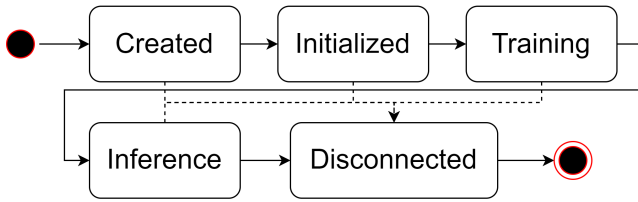


Fig. 1: The state machine representation of a session

but no communication occurred, and the server is waiting for a client to connect.

The client sends an *InitRequest* message to the server to establish a new session.

After initialization, the client may proceed to the states *Training* or *Inference*. If the client establishes a new session with a server, the user has to train the classifier model, and the session must proceed to the *Training* state. If a context with a trained model can be restored for the user (e.g. in case of reconnection caused by a network error) the *Initialized- i* , *Inference* state transfer is legal and should be used.

In both *Training* and *Inference* states the protocol transmits and accepts messages related to EMG processing. Data-related communications happen asynchronously, and data messages may not alter the state of either party.

The client may terminate the session in any state. When the server receives a *DisconnectRequest*, the server informs the client that the session state is *Disconnected*, the session is terminated, and all server-side resources related to the session are released.

The protocol uses 32-bit IDs that wrap on overflow by design. This ID length is used, as even at a relatively high (1000 messages/s) message rate, it would still take the counters more than 1000 hours to wrap. The sequence numbers provided by the DTLS layer are not used (in upper layers), as *NWE mg* sessions are not bound to DTLS sessions.

VI. BENCHMARKS AND TESTS

To minimize the skew of results caused by resource constraints of different configurations, devices used in each case were significantly more powerful than what would be required during the regular operation of the protocol.

Tests used a PC equipped with an Intel Core i5-10400 CPU, 32 GB of 3600 MHz DDR4 memory, a Realtek RTL 8111 GbE Network controller for measurements in which ethernet was used to connect to a Customer Premise Equipment (CPE) and an Intel Dual Band Wireless-AC 3168NGW WiFi 5 adapter used for Wi-Fi 5 measurements

For monitoring purposes, a different network interface controller (NIC) was used, and the routing table was set up for the client to use the specified interfaces only for traffic relevant to the measurements.

In the case of measurements that featured a phone, we used a Poco X6 5G⁶. We used UserLAnd⁷ to run Ubuntu on the phone, and the protocol implementation was compiled to AArch64 native binaries.

The server we used was a 5G MEC application server, with 16 GB of memory and 2 Virtual CPU cores.

Every test was compiled with release-mode⁸ Rust 1.81 and Tokio 1.40, benchmarks were executed as unit tests, and the

server was executed as a regular process. All processes were (re)started for each test to avoid state or counters skewing the result.

Ubuntu 24.04 was used on clients and Ubuntu 22.04 LTS on servers. Windows devices were also used both as a client and as a server to detect platform-specific behaviour, but only Linux-based measurements are shown for consistency. The performance differences between platforms were negligible.

Measurements that use the *NWE mg* protocol use DTLS as per the specification, even in *measure* mode. Tests were executed with varying payload sizes and different message rates, on the BME-Nokia 5G research network⁹ (5G SA) with us being the only user of a cell, the One network (5G NSA), eduroam Wi-Fi 5 and Wi-Fi 6 APs with average background load. The eduroam network had direct access to the MEC host.

The protocol relies on asynchronous tasks, thus a purely synchronous application that serves as the control in each test environment was desirable to verify or disprove theories regarding packet ordering, queuing and the impact of DTLS. A simple synchronous echo utility that transmits messages of the *NWE mg* protocol was created that helped us discover the extreme impact of queuing in scheduled networks which (in extreme cases) resulted in the asynchronous code dropping 60-70% of the messages due to a newer message being received by the time the messages pass DTLS decryption.

To validate the results measured using our custom tools, we ran tests with IPerf and used the Linux ping command with parameters set as close to the protocol's parameters as possible. With both utilities, we managed to produce results within an acceptable margin of error.

To simplify benchmarking the protocol, a measurement module was developed along with the rest of the native code base. The module collects all results in in-memory data structures (in the current implementation a pre-allocated vector with a capacity of 100 000 log items) to minimize the impact of data collection. At the end of each session, a CSV version of these structures is written into a timestamped log file, prefixed with the name of the test performed. The module provides various benchmarking features with a configurable message size, frequency, and data source (e.g., randomly generated or imported real-world data).

VII. RESULTS

The results presented in the following subsections are based on measurement sessions for each parameter set and access network with 100,000 data messages sent.

In each test case, the same set of graphs (scatterplots and empirical distribution function (ECDF) diagrams) showing the top 99%, top 95% and top 90% data and tables detailing RTT and delivery statistics were generated, available at GitHub.

The overhead of the protocol and the network stack without physical media and with Ethernet was measured with the protocol running on localhost, and through Ethernet, as Ethernet was used to connect the PC to the 5G CPEs. We measured an average RTT of about 0.3 ms using localhost, and 0.5 ms over an Ethernet connection.

Even though the technologies shown are significantly different in their characteristics, 200 messages/s seemed to be

⁶https://www.gsmarena.com/xiaomi_poco_x6-12723.php

⁷<https://github.com/CypherpunkArmory/UserLAnd>

⁸<https://doc.rust-lang.org/cargo/reference/profiles.html#release>

⁹<https://5g.bme.hu/>

the optimal spot for most technologies. This is also the expected message rate in a real-life setting. Surprisingly, no technology did significantly worse at 200 messages/s than at lower rates, and in most cases, lower than 200 messages/s message rates yielded worse results. The 5G SA network had no issues keeping up with increasing message rates, with the latency measured being affected mostly by scheduling rather than limited capacity while both Wi-Fi standards started to struggle at 1000 messages/s.

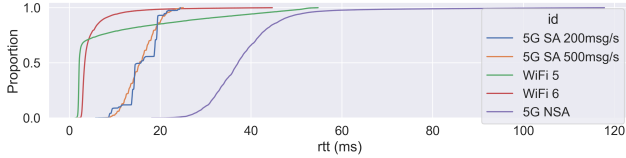


Fig. 2: All technologies compared (ECDF, top 99%)

While both Wi-Fi standards provided surprisingly low average latencies (usually well below 10 ms, Wi-Fi 6 had an average top 95% RTT of 4.1 ms with maximum message size), the mean jitter was (relatively) higher, and 5G networks were more consistent (i.e. the minimum and maximum latencies were much closer) as shown on Fig. 2 while having higher mean RTTs, starting at about 16 ms.

In the case of scheduled networks, out-of-order or grouped reception of packets was more prevalent than we anticipated.

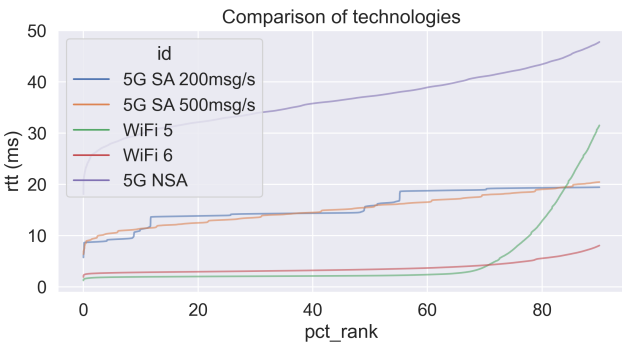


Fig. 3: All technologies compared (Rank to RTT (ms), top 90%)

Fig. 3 shows the top 90% data so that differences beyond Wi-Fi 5 being inconsistent are visible. All wireless technologies we benchmarked handled the measurements without losing connection, and even the worst results are good enough that we feel confident that under ideal conditions all access technologies are capable of handling the task.

Our tests showed that Wi-Fi 6 (along with later versions of Wi-Fi) might be a viable alternative to 5G NR inside buildings where 5G has worse coverage, and in areas where 5G coverage is not yet available but low-latency access to a MEC host is feasible. 5G provided profound consistency and range during our measurements and is advantageous as it has proper support for mobility such as reliable handover mechanisms.

VIII. CONCLUSIONS AND FUTURE WORK

Intelligent prostheses that offload the processing of EMG data rather than processing measurements locally could provide a better user experience, as algorithms executed by edge cloud solutions are far less constrained in complexity and energy requirements. A protocol and a framework were developed to facilitate the offloading of EMG processing.

While most carriers are yet to deploy 5G MEC servers in their networks, most multinational carriers already market MEC capabilities as part of their offerings, and the technology is the subject of many active research projects. The demand for MEC deployments should increase substantially with the appearance of new applications that directly benefit from predictable (low) latency or high bandwidth. Our framework also provides benchmarking utilities that can help determine the viability of offloaded processing in certain networking environments. We conducted extensive benchmarks and experiments on several networks in various contexts that helped us better understand the characteristics and behaviour of different wireless networking technologies and edge cloud solutions. Based on these benchmarks, we can conclude that modern wireless networking technologies in combination with lightweight domain-specific protocols are capable of transmitting data reliably enough to be suitable for offloading control-related computations in situations, where an end-to-end latency in the order of or below the fastest human reaction time (about 100-150 ms) is acceptable.

The results presented in this paper are universally applicable to other domains with similar latency and bandwidth requirements. Even in less time-critical applications benefits might come with offloading computations, the most prominent being the availability of significantly more powerful computational resources with virtually no energy constraints.

The protocol we created is a minimal-overhead, relatively simple solution for handling low-volume, time-critical and domain-specific communication complete with EMG processing. With the incorporation of 5G and B5G technologies such as the Radio network information service (RNIS) [17] and QoS flows [18] could ensure the predictable delivery of EMG data, and the N3IWF [19] provides a standardized way of incorporating access to the 5G core network (and as such, MEC) through non-3GPP access networks¹⁰. Using the MEC orchestrator, we can ensure that the application is deployed on a MEC host that provides the least latency on a per-user basis, and ensures proper scaling if required. The upper bound we have set in Goals was not only met but according to the measurements, a significant part of the targeted e2e latency remains available for server-side processing.

Further research is required to uncover the exact accuracy gain this solution could provide, and whether the data collected during normal operations could be subjected to offline analysis to further improve efficiency. Ultimately, controlling intelligent prostheses can be and hopefully will be one of the prime use cases for 5G, B5G and edge cloud services. The project has reached the stage in which the next steps are the deployment of the solution in an experimental physical setting (i.e. on a physical prosthesis connected to the MEC application through 5G or Wi-Fi) and benchmarking and testing the accuracy and performance of different classifier models be executed by the server. The current implementation of the server retains the data received for logging and analysis. Further research is required to determine whether the data collected could be used to increase accuracy. After further testing of the protocol while running algorithms, the protocol will be deployed to a physical prosthesis. During the first round of testing the client will run on an external device, but later, a 5G-capable embedded client might be used.

¹⁰<https://www.linkedin.com/pulse/n3iwf-mec-gpon-5g-abu-hayat-khan/>

The implementation of the protocol is not yet publicly available as it is subject to IP-related procedures and major changes are expected throughout the initial deployment process.

ACKNOWLEDGMENT

This research was supported by the Ministry of Culture and Innovation and the National Research, Development and Innovation Office within the Cooperative Technologies National Laboratory of Hungary (grant No. 2022-2.1.1-NL-2022-00012).

REFERENCES

- [1] E. G. M. 003, "Multi-access edge computing (mec); framework and reference architecture," Technical Specification (TS), 2022. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/03.01.01_60/gs_MEC003v030101p.pdf
- [2] O. Samuel, M. Asogbon, Y. Geng, A. Al-Timemy, S. Pirbhulal, N. Ji, S. Chen, P. Fang, and P. Li, "Intelligent emg pattern recognition control method for upper-limb multifunctional prostheses: Advances, current challenges, and future prospects," *IEEE Access*, vol. PP, pp. 1–1, 01 2019.
- [3] J. LEÓN-PERALTA, R. Sanchez-Lara, J. Vazquez-Avila, and J. YAÑEZ-VARGAS, "Implementation of a neural network of low computational cost for its application in arm prostheses," *Revista de Ingeniería Tecnológica*, pp. 27–34, 11 2022.
- [4] N. Witt, M. Deutel, J. Schubert, C. Sobel, and P. Woller, *Energy-Efficient AI on the Edge*, 07 2024, pp. 359–380. [Online]. Available: https://www.researchgate.net/publication/382679690_Energy-Efficient_AI_on_the_Edge
- [5] G. Nencioni, R. Garroppo, and R. Olimid, "5g multi-access edge computing: A survey on security, dependability, and performance," *IEEE Access*, vol. PP, pp. 1–1, 01 2023.
- [6] A. Filali, A. Abouaoumar, S. Cherkaoui, A. Kobbane, and M. Guizani, "Multi-access edge computing: A survey," *IEEE Access*, vol. 8, pp. 197 017–197 046, 2020.
- [7] E. W. P. N. 55, "Mec support towards edge native design," Tech. Rep., 2023. [Online]. Available: https://www.etsi.org/images/files/ETSIWhitePapers/ETSI-WP55-EC_support_towards_Edge_native.pdf
- [8] M. Ghoshal, P. Dash, Z. J. Kong, Q. Xu, Y. Hu, D. Koutsonikolas, and Y. Li, *Can 5G mmWave Support Multi-user AR?*, 01 2022, pp. 180–196. [Online]. Available: <https://repository.library.neu.edu/files/neu:4f17qg40k/fulltext.pdf>
- [9] A. Aijaz, "Private 5g: The future of industrial wireless," *IEEE Industrial Electronics Magazine*, vol. 14, no. 4, pp. 136–145, 2020.
- [10] N. Torrisi, J. Meira, G. Matos, A. Perdigão, J. Cação, C. Resende, W. Moreira, M. Antunes, J. Quevedo, R. Moutinho, J. Oliveira, P. Rendeiro, P. Oliveira, A. Oliveira Junior, J. Santos, and R. Aguiar, "Industrial internet of things over 5g: A practical implementation," *Sensors*, vol. 23, 05 2023.
- [11] Y. Liu, M. Peng, G. Shou, Y. Chen, and S. Chen, "Toward edge intelligence: Multiaccess edge computing for 5g and internet of things," *IEEE Internet of Things Journal*, vol. 7, pp. 6722 – 6747, 06 2020.
- [12] E. J. Scheme and K. B. Englehart, "Electromyogram pattern recognition for control of powered upper-limb prostheses: state of the art and challenges for clinical use." *Journal of rehabilitation research and development*, vol. 48 6, pp. 643–59, 2011. [Online]. Available: <https://api.semanticscholar.org/CorpusID:14883575>
- [13] N. Parajuli, N. Sreenivasan, P. Bifulco, M. Cesarelli, S. Savino, V. Niola, D. Esposito, T. J. Hamilton, G. R. Naik, U. Gunawardana, and G. D. Gargiulo, "Real-time emg based pattern recognition control for hand prostheses: A review on existing methods, challenges and future implementation," *Sensors*, vol. 19, no. 20, 2019. [Online]. Available: <https://www.mdpi.com/1424-8220/19/20/4596>
- [14] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," RFC 7252, Jun. 2014. [Online]. Available: <https://www.rfc-editor.org/info/rfc7252>
- [15] M. Scharf and S. Kiesel, "Head-of-line blocking in tcp and sctp: Analysis and measurements," 01 2007, pp. 1 – 5. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/4150963>
- [16] L. Eggert, G. Fairhurst, and G. Shepherd, "UDP Usage Guidelines," RFC 8085, Mar. 2017. [Online]. Available: <https://www.rfc-editor.org/info/rfc8085>
- [17] E. G. M. 012, "Multi-access edge computing (mec); radio network information api," Technical Specification (TS), 2019. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/MEC/001_099/012/02.01.01_60/gs_mec012v020101p.pdf
- [18] 3GPP, "5G System; Policy and Charging Control signalling flows and QoS parameter mapping; Stage 3," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 29.513. [Online]. Available: <http://www.3gpp.org/DynaReport/29513.htm>
- [19] —, "Access to the 3gpp 5g core network (5gcn) via non-3gpp access networks," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 24.502. [Online]. Available: <http://www.3gpp.org/DynaReport/24502.htm>