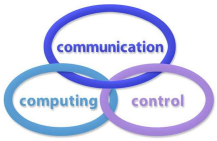
INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
Online ISSN 1841-9844, ISSN-L 1841-9836, Volume: 15, Issue: 4, Month: August, Year: 2020
Article Number: 3842, https://doi.org/10.15837/ijccc.2020.4.3842



CCC Publications

Proposed Fuzzy Real-Time HaPticS Protocol Carrying Haptic Data and Multisensory Streams

S. Kontogiannis, G. Kokkonis

Sotirios Kontogiannis*
Department of Mathematics
University of Ioannina, Greece
Ioannina, 45100, Greece
*Corresponding author: skontog@cc.uoi.gr

George Kokkonis
Department of Business Administration
University of Western Macedonia, Greece
Grevena, 46100, Greece
kokkonisgeo@gmail.com

Abstract

Sensory and haptic data transfers to critical real-time applications over the Internet require better than best effort transport, strict timely and reliable ordered deliveries. Multi-sensory applications usually include video and audio streams with real-time control and sensory data, which aggravate and compress within real-time flows. Such real-time are vulnerable to synchronization to synchronization problems, if combined with poor Internet links. Apart from the use of differentiated QoS and MPLS services, several haptic transport protocols have been proposed to confront such issues, focusing on minimizing flows rate disruption while maintaining a steady transmission rate at the sender. Nevertheless, these protocols fail to cope with network variations and queuing delays posed by the Internet routers.

This paper proposes a new haptic protocol that tries to alleviate such inadequacies using three different metrics: mean frame delay, jitter and frame loss calculated at the receiver end and propagated to the sender. In order to dynamically adjust flow rate in a fuzzy controlled manners, the proposed protocol includes a fuzzy controller to its protocol structure. The proposed FRTPS protocol (Fuzzy Real-Time haPticS protocol), utilizes crisp inputs into a fuzzification process followed by fuzzy control rules in order to calculate a crisp level output service class, denoted as Service Rate Level (SRL). The experimental results of FRTPS over RTP show that FRTPS outperforms RTP in cases of congestion incidents, out of order deliveries and goodput.

Keywords: multi-sensorial, haptics, transport protocols, real-time protocols, communication networks, fuzzy logic based decisions.

1 Introduction

Real-time multi-sensory and multi-control actuating protocols try to transfer sensory and control haptic data over the Internet as a unified flow content profile. As the Internet is transforming towards the *"Internet of Everything"*, multi-sensory data profiles are massively utilized, characterized with low bit-rates, synchronized back to back frames of minimum jitter and zero packet loss tolerance [3].

Existing transport protocols and mainly TCP protocols focus on big data payload transfer sizes usually above 512 bytes. TCP protocols most important functionality is congestion control, performed using congestion window (cwnd) variations [18]. Congestion window as well as other conservative TCP friendly mechanisms such as congestion avoidance, fast recovery and slow start, force TCP protocols to operate in packet bursts, introducing unnecessary jitter for real-time flows [4]. Furthermore, TCP varieties set as packet loss indication the existence of duplicate ACKs or timeouts without accurate probing and continuously monitoring network conditions, so as to adjust the cwnd accordingly [1]. Such real-time detection inaccuracies left space for other unreliable but of constant rate protocols like UDP to prevail over TCP for real-time applications.

Several forms of real-time protocols descendants from UDP are presented in the following literature section. However, the mostly used representative with the highest applicability is the Real Time Protocol (RTP) [26]. The RTP tries to maintain a steady rate as UDP, while it provides both flow control and a re-transmission mechanism to compensate for packet losses. It is also augmented by another protocol called RTCP which provides control and timely feedback to both communicating participants such as the inter-arrival jitters and packet losses.

Usually multi-sensory data flows are coupled together with audio-video real-time data flows coming from a single or multiple data streams of significant data volume. Such flows are characterized by low and constant bit rates and small payload sizes (not more than 350bytes). These flows are traversing as part or independently along with a media streaming protocol over the Internet. In case of congested links, such flows are exposed to significant delays and jitter that in turn stress out significantly the real-time synchronization and functionality of the application layer [12]. According to [2] humans have limited adaptability to time delay that varies between 0.3-0.5sec, while data loss is one of the most critical problems to be particularly solved by constant bit rate sources. Furthermore, adaptation to the Internet characteristics is an important factor that needs to be taken into consideration in order to develop reliable Internet-based real-time protocols for control systems [32].

For real-time systems, quality of Experience (QoE). QoE is the measurement of perception of a real-time application by the end user. Several QoE measures have been proposed such as goodput, average throughput re-buffering, start-time, bit-rate shifts (jitter), data resolution and service response time, trying to measure of real user quality indexes such as ITU application rating, Mean Opinion Score (MOS), application's performance, controller capabilities, data communication costs, service failure incidents, et.al [23, 47]. For teleoperations, all of the aforementioned measures are used for probing real-time application delays and loses, while offering service transparency and application stability. Recent surveys show that 60% of all video streams in the Internet suffer from poor quality [43] and 40% of all consumers are concerned about the Internet video quality [44]. Such poor performance of Internet based real-time haptic and sensory applications leave space for more adaptive protocol design and intelligent approaches.

This paper includes a new proposition of a fuzzy transport protocol for real-time multi-sensory applications. Its implementation is instructed by the next generation networks and IoE requirements. For instance, the pico-cells deployment and escalation of 5G infrastructure and the need for high availability and better than best effort wireless tactile services. Service supporting data augmentation [28], virtual reality [58], medical tele-operations [20, 30, 41], automated vehicles [52] and industry 4.0 applications [37]. This paper describes a new protocol caller FRTPS (Fuzzy Real-Time haPticS protocol), that includes a fuzzy control component that uses receiver-end calculated measurements of Packet Arrival Deviations (PAD), Mean ACK time Delays (MD) and packet loss (PLOS), in order to detect the network contention or congestion incidents and respond with either flow rate reduction or data quantization. Moreover, scenarios experimentation and protocol evaluation results follow in terms of both performance and quality of experience.

This paper is divided as follows: In section 2 related work on existing real-time protocols carrying

sensory data are presented. Section 3 presents the authors' proposition of the FRTPS protocol and functionality. Section 4 includes the FRTPS experimentation and experimental results, while section 5 concludes the paper.

2 Related work on existing sensory protocols

In order to transfer multi-sensory data through the Internet, apart from TCP, UDP and RTP [26], several real-time transport protocols operating in accordance with the carried data requirements have been introduced. Several protocols have been developed for this reason specifically for haptic applications that control robotic arms, haptic gloves that receive vibration feedback, glasses and tablet applications that project augmented sensory information into things [7]. The most important representatives are outlined in the paragraphs that follow.

Stream Control Transmission Protocol (SCTP) is a TCP variant for real-time streams. It has the ability to aggregate different streams into one message flow (multi streaming). SCTP data chunks are identified by three items: TSN, SI and SSN. Transmission sequence Number (TSN) is a cumulative sequence number, different for each stream chunk in the message; Sequence Identifier (SI) is the stream identifier and Stream Sequence Number (SSN) to distinguish different data chunks in a message that belong to the same stream [54]. In SCTP control and data information are also carried in the same message, thus making SCTP as the best candidate for SIP and VoIP(H.323) data deliveries[21]. SCTP uses the same congestion control and flow control mechanisms with the TCP protocol, which means that is lacks of a real-time network conditions measurement mechanism and it carries aggregated flow chunks which in turn may cause multi stream disruption in case of congestive incidents, especially in transient loses or random drops. This leads to poor performance in wireless and ad hoc environments [36].

Moreover, the Datagram Congestion Control Protocol (DCCP) is also is a better improvement of SCTP, that incorporates the Explicit Congestion Notification (ECN) mechanism and can also serve for reliable in order delivery for UDP segments [10, 16], As compared by [5], it outperforms SCTP in 4G environments, but still utilizes the TCP congestion control mechanisms drawbacks for real-time content deliveries. Another TCP variant is the The Interactive Real Time Protocol ($IRTP^+$).

Synchronous Collaboration Transport Protocol ($SCTP^+$) is an interactive stream protocol that transfers stream data over UDP. It has many similarities to the RTP such as message sequence numbering (flow control), and stream id (without multi stream support as SCTP does). $SCTP^+$ uses a delayed Negative Acknowledgment mechanism to indicate packet losses similar to the RTCP protocol (part of the RTP loss report mechanism) [15]. Nevertheless, $SCTP^+$ includes a mechanism, when multiple recipients (recipients group) are involved for the sender to multicast its messages while receiving ACKs for each message only from the recipient of the group that the initial NACK mechanism indicated the highest RTT value [49]. An improved version of $SCTP^+$ for haptic applications is smoothed SCTP (S-SCTP), that adds a jitter smoothing mechanism of a receiver buffer at the $SCTP^+$ protocol [13]

The ALPHAN Protocol operates on top of UDP and it is used by Collaborative Haptic Audio Visual Environment (C-HAVE) technologies in virtual reality, surgical simulations and games [42]. It is mainly an application layer protocol that includes flow control, messages re-transmission, and embedded timestamp on its messages headers, similar to the RTP protocol.

Interactive Real-Time Protocol (IRTP) is a minimum header RTP like protocol that has been designed for interactive internet-based services on top of IP. In order to reduce the end-to-end delay IRTP uses the TCP handshake and connection establishment mechanism, and adopts TCP congestion window. However, in contrast to TCP, IRTP sender increases its packet in flight as well as the RTP flow control,and re-transmission mechanism. IRTP protocol ACKs include a available receiver *rwnd* field to inform the sender of its goodput availability. In cases of difference between flow goodput and throughput, the IRTP enters its congestion avoidance mechanism by multiplicative reducing its frame rate by some proportion: $W_{t+\partial t} = (1 - \beta)W_t$ [34].

Bidirectional Transport Protocol (BTP) is an alternative protocol to IRTP that focuses on reducing the end-to-end delay monitoring by introducing the IPG (Inter Packet Gap) metric of time interval

between two successive data packets or frames and the [59, 60]. This IPG-based control mechanism provides congestion control similar to the TCP window size based congestion control. BTP is UDP based that in cases of network congestion attempts to provide a uniform end-to-end delay by modifying data packets in flight based on receiver IPG feedback.

Furthermore, other less important real-time protocols such as the Efficient Transport Protocol (ETP), Real Time network Protocol for Interactive media (RTP/I), Real Time Network Protocol (RTPN), Hybrid Multicast Transport Protocol (HMTP), Scalable Reliable Multicast (SRM), Reliable Multicast Transport protocol (RMTP), Selective Reliable Transmission Protocol (SRTP) are outlined at [31].

Since existing systems OS kernels can tolerate correlated functions and operations to reach decisions or deduct conclusions from past data very fast, sophisticated monitoring algorithm implementations are underway such as the introduction of fuzzy detectors to the TCP congestion avoidance mechanism with adaptive escalation strategies [6, 39, 55]. Furthermore, it is also mandated by the failure of existing TCP congestion mechanisms to deliver accurate flow rate requests [29].

Providing adaptive to conditions networking protocols over TCP is an ongoing research effort [39]. Moreover, the existing implementations towards this direction are the FL-TCP; a TCP Vegas based protocol, that imposes a Fuzzy controller over expected and actual RTT calculations at the transmitter end [51], FLOWER [55], that provides an RTT fuzzy controller for BitTorrent clients and Fuzzy TCP. The main issue with these TCP protocol implementations (apart from FLOWER), is that they provide a more of generic adaptive congestion window (cwnd) increase or decrease, based on RTT measurements only, without differentiating protocol use according to each application profile requirements.

The proposed FRTPS protocol in this paper is an RTP based protocol, that carries some of the layer adaptation attributes of AMESETP protocol. AMESETP is part of MESETP protocols suite for real-time medical services. AMESETP protocol monitors packet loss and inter-frame delay variation at the receiver end. It then signals the sender via ACK frames to either increase or decrease its frame rate by reducing or increasing the number of bytes/frame accordingly. AMESETP offers 7 levels of rate service while some of them offer also quantization (compression of payload data) [30]. As examined by the AMESETP authors, the AMESETP decisions have shown significantly better performance in comparison to UDP, close to performance of the RTP protocol [30]. Nevertheless, AMESETP lacks of a fast adaptive mechanism, and requires a bigger number of service rate levels as well as a more accurate network conditions probing mechanism, in order to outperform RTP. Proposed FRTPS protocol and functionality is presented at section 3.

3 Proposed FRTPS protocol

FRTPS (Fuzzy Real-Time haPticS) protocol is a new proposed transport protocol for small payload, low bit-rates, real-time flows. FRTPS includes a new flow control mechanism of continuous network probing and fuzzy logic rate adaptation. The motivation for a fuzzy transport protocol derives from [9, 19, 45], that tried to deal with robot movement uncertainties, implementing their proposition at the application layer. The proposed FRTPS protocol is more of a generic approach that can enforce policies based on network conditions, thus making it more general and scalable for sensory, actuator data transmissions.

FRTPS adaptation policy is based on a a fuzzy estimator on crisp input measurements performed by the receiver end. FRTPS does not include TCP like congestion avoidance and rate adaptation mechanisms, apart from the mechanism used by RTP and utilizes only its adaptive fuzzy rate-quantization escalation logic. As RTP protocol dictates, FRTPS also includes a timestamp field embedded in its frames so as to assist application layer protocols frame synchronization.

FRTPS protocol description is divided as follows: Section 3.2 presents the FRTPS metrics used by its fuzzy controller to calculate the offered service level. Section 3.2 presents the FRTPS protocol structure and rate adaptation layers, while section 3.3 presents in detail the FRTPS fuzzy control layer functionality.

3.1 FRTPS metrics

FRTPS aggregates, quantizes or both its data transmissions, based on three recorded metrics at the receiver end:

1. Packet loss (PLOS) measurements at the receiver end, smoothed over an EWMA process (see Figure 2)
2. Packet Arrival Deviation (PAD) measured by the receiver flow monitoring agent (see Figure 2)
3. Mean packet delay metric (MD), supported by the FRTPS frame timestamp field and accurate synchronization between sender and receiver using either Precision Time or Network Time Protocol (NTP) (MD is measured by the sender flow monitoring agent, see Figure 2).

All the above metrics are used by the FRTPS protocol fuzzy controller as input for the calculation of a single crisp output that corresponds to a specific rate adaption layer. Detailed metrics analysis follows.

Packet loss (PLOS) metric is also used by the FRTPS protocol and adapts to the FRTPS protocol an EWMA process for the next time interval packet loss estimation. The estimator equation used for k+1 interval is the presented in Eq. 1:

$$PLOS_{k+1} = \frac{\beta}{k-2}\sum_{i=1}^{k-1} PLOS_i + (1-\beta)\overline{PLOS}_k \tag{1}$$

where k is the time depth (k>2) of the EWMA process and coefficient β value is calculated as follows:

$$\beta = 0.2 + \max_{i=1}^{k}(PLOS_i) - \min_{i=1}^{k}(PLOS_i) \tag{2}$$

PLOS metric is calculated at the receiver and it is sent back to the sender using appropriate FRTPS acknowledgment packet packet loss field. This information is delivered to the sender in order to adjust its frame rate after the FRTPS adaptive fuzzy estimator. PAD metric is another metric partly calculated at the receiver end as the std. deviation of each received packet and sent back to the sender via appropriate ACK frame field. The sender then subtracts the deviation of each previous received ACK packet.

Packet Arrival Deviation (PAD) is a metric that expresses the delay deviation (jitter) of a specific period between two consecutive numbered frames. Prior to PAD presentation a short analysis of relevant measures follows.

Inter packet delay variation is the time variation between two consecutive packets i, i+1 is expressed as: $IPDV = D_{i+1} - D_i = Rt_{i+1} - Rt_i + Tt_{i+1} - Tt_i$, while packet delay variation is expressed as: $PDV_i = D_i - D_{min} = (Rt_i - Tt_i) - D_{min}$, where Rt is the reception time as recorded by the NTP source and Tt is the transmission time as recorder by the same NTP source (It is significant that both sender and receiver have similar stratum values from the NTP source). IPDV is a measure of the network's ability to preserve the spacing between packets, while it reduces the demands on the stability and skew of measurement clocks [40]. PDV on the other hand can not distinguish short term variation, it is sensitive to the first frame delay and strong correlated with the mean frame delay (MD). Another metric value is the Mean Absolute Packet Delay Variation (version) 2 (MAPDV2), and is specified in as it computes a smoothed running estimate of the mean delay using the one-way delays of 16 previous packets [11, 50]:

$$\begin{aligned} MD_{n=1..16} =& a_n = \frac{1}{16}D_{n-1} + \frac{15}{16}a_{n-1} \\ if \quad & t_{n+i} > a_n, P(n+i) = P(n+i) + (t_{n+i} - a_n) \\ if \quad & t_{n+i} < a_n, N_{n+i} = N_(n+i) + (a_n - t_{n+i}) \\ MAPDV_2 =& \bar{P} + \bar{N} \end{aligned}$$

RTP Jitter J_i metric is used by RTP and it expresses the undesired deviation of a true periodical flow and is described according to [26], for an RTCP interval between packet i and packet j as:$D(i,j) =$

$(R_j - R_i) - (S_j - S_i) = (R_j - S_j) - (R_i - S_i)$ and $J_i = \frac{d(R_i - S_i)}{di}$ and estimated using the Jacobson estimator [25] as: $J_i^{est} = J_{i-1} + (|D_{(i-1,i)}| - \frac{J_{i-1}}{16})$, expressed by RTP as inter-arrival jitter [40]. Moreover, jitter for elongated intervals called jitter buffers capable of maintaining n packets is calculated as the standard deviation of the mean $\frac{RTT_i}{2}$ time of a packet leaving the sender until an ACK packet is received by the sender: $J_k = \sqrt{\frac{\sum_{i=1}^{k} (\frac{RTT_i}{2} - \frac{\bar{RTT}}{2})^2}{n-1}}$.

The PAD metric is expressed as follows: At first packets i, i+1 inter-arrival time is calculated ($D_i' = \frac{dR_i}{di}$) at the receiver end. Then, according to Equation 3 the receiver pad value is calculated. This value is send back to the sender using appropriate ACK FRTPS frame field (see Figure 1, D_i' receiver inter-packet variance field of FRTPS protocol ACK header).

$$PAD_i^{R_x} = \sqrt{(D_i')^2 - (\bar{D}')^2} \tag{3}$$

where $\bar{D}'$ is the mean value calculated by the receiver. The sender also makes its own variance calculation based on i, i+1 ACK packets reception, of the incoming ACK packets based on Equation 4:

$$PAD_i^{T_x} = \sqrt{(D_i'')^2 - (\bar{D}'')^2} = \frac{k-1}{\frac{T_{int}}{p_{agg}}} \tag{4}$$

where $D_i'' = \frac{dR_i^{ACK}}{di}$ and $\bar{D}''$ is the inter arrival time of received i, i+1 ACK packets at the sender. The previous equation can be further simplified, where k is the back to back frame re-transmissions due to duplicate ACKS (RTP congestion avoidance mechanism embedded to FRTPS) and T_{int} is the current inter package(frame) gap (IPG_i) denoted by the sender's frame rate and p_{agg} is the number of aggregated packets per frame. The total PAD calculation done by the sender based on Equation 5:

$$PAD_i = \sqrt{\frac{1}{2}(PAD_i^{R_x})^2 + (PAD_i^{T_x})^2} \tag{5}$$

According to Equation 5, PAD values are greater than zero, the root mean square offers a smooth linear approximation over non linear abrupt variations. Such an approximation is useful and less time consuming for FRTPS that relies its adaptation on complex fuzzy calculations. PAD metric does not depend on the FRTPS flow control mechanisms. This means that it may give measurements based even on out of band packets received at the receiver end. In case of packet drops, PAD values increase close to the re-transmission timeout or to k times the periodic frame sent interval T_{int}, until a new acknowledgment frame is received. if $PAD > T_{int}$ then PAD is set to T_{int} value ($PAD = T_{int}$). This makes PAD over time a smooth curve towards abrupt short duration jitter intervals, as well as sensitive in case of congestive packet loss incidents (more than one-burst packet losses).

MD metric is calculated as the time difference between packet transmission and acknowledgment reception at the sender. That is, an approximation of instant MD metric is: $MD_k = \frac{RTT_k}{2}$. FRTPS calculates MD as a quadratic moving average and provides the flow with stochastic next time interval MD prediction. Under no congestive or error conditions, MD has a smooth quadratic response up and down a mean value. In case of congestive saturated links, depending on the communication channel congestion incident and its participating flows immediate or delayed avoidance, MD calculus quadratic response values are smoothly exponentially increased depending on the packets delay effect over the last i packets . FRTPS MD metric for the next time interval (k+1) is calculated as follows:

$$MD_{k+1} = \alpha \sum_{i=1}^{k} \frac{1}{(k-i)^2} + (1-\alpha) MD_k - \epsilon_k \tag{6}$$

where k is the time depth coefficient ($k \cdot T_{int}$) that expresses the number of previously consecutive transmitted packets ($i = 1..k, k \geq 2, k \geq 1$) of the stochastic process maintained by each FRTPS flow. MD_k value is the mean delay of the last transmitted packet while MD_{k+1} is the predicted mean delay of the next packet in the transmission queue. Coefficient a, is calculated according to the following

formula:

$$\alpha = \sum_{i=1}^{k} \frac{1}{(k-i)^2} < 0.7 \tag{7}$$

From Equation 7, as k increases (100<k<1000), parameter α can be approximated to $\alpha = \frac{2}{3}$. For k values less that 10, parameter α is approximated to $\alpha = \frac{1}{2}$. Error parameter ϵ_k is calculated as the difference: $\epsilon_k = MD_k^{predicted} - MD_k^{real}$.

3.2 FRTPS protocol structure

FRTPS is a middle-ware protocol that carries sensory measurements or haptic equipment real-time control operations. FRTPS frame structure is presented at Figure 1 and includes the following fields: A sequence number based on UTC timestamps for flow control, a Sending Rate Level (SRL) field, calculated and filled by the sender, an Equipment ID field that uniquely identifies each flow, a length (LEN) field that represents the number of sensor values included in the payload (max of 16 sensor values can be included per packet) and the packet sensor payload of 4 octets for each sensor value. Finally, an aggregation id field is used (Agg. ID), that identifies the number of aggregated packets included per FRTPS frame. Each frame is transmitted from its source periodically, using a constant frame transmission interval of: 5ms<T_p<60ms. For each frame or grouped packets received by the receiver, an acknowledgment frame is send back to the sender. ACK frame structure is illustrated at Figure 1.

ACK frames include fields such as the SRL(Service Rate Level), frame LEN, Equipment ID and Aggregation Agg. ID of the corresponding source frames. LEN and Agg. ID fields are included as a validation of packet reception payload and/or packets, while SRL is included in order to indicate that SRL change has been propagated to the receiver in case of SRL rate adaptation. ACK packets also include a PLOS metric estimation for the PLOS metric as well as an inter-packet variance estimation Di' value and used by the transmitter in order to calculate PAD metric value and to adapt its rate for the following time interval accordingly. FRTPS protocol does not include a native re-transmission mechanism nor does it uses RTO intervals for timeout or frame RTR purposes. It keeps on transmitting packets even if packet losses are experienced. That is why PLOS metric is calculated at the receiver end, which has the complete information on what packet has been received.

Bits	***1-4***	***5-8***	***9 – 16***	***17-24***	***25-32***
0	Sequence Number-UTC time based				
32	SRL	LEN	Agg. Id	Equipment ID	
	Sensor 1 value				
	Sensor 2 value				
	…				
	Sensor 16 value				

Bits	***1-4***	***5-8***	***9 – 16***	***17-24***	***25-32***
0	Ack Number				
32	SRL	LEN	Agg. Id	Equipment ID	
	PLOS estimated value				
	Di' receiver inter-packet variance				

Figure 1: FRTPS middle-ware protocol header and FRTPS delayed ACK frame structure

FRTPS adaptive rate mechanism includes 13 rate-levels called Service Rate Levels or SRLs and are illustrated at Table 1. Each SRL level has a service id, and service attributes that correspond to: 1.The number of packets encapsulated per frame, and 2.The per sensory value quantization properties of 32 to 32, 32 to 24, 32 to 16 and 32 to 8 bits per sensory payload data value compression (quantization).

Starting from the default SRL=1 level, each source flow periodically sends FRTPS packets to the receiver and awaits acknowledgments. If the FRTPS moves to a higher SRL level, it uses the new SRL packet aggregation or quantization attributes for the next transmitted frames, propagates the new SRL level to the receiver and confirms the SRL level change by the SRL field of the next incoming ACK frame. FRTPS service levels cope with the following policy regarding real-time traffic:

> "If rate increase is required due to limited BW resources, then at first increase packets per frame using packet aggregation and then reduce frame packed payload values resolution".

Table 1: FRTPS protocol fuzzy partitioning output rate levels (Service Rate Levels) and their corresponding attributes.

SRL	SRL level name	Aggregated sensor values/sensor field/frame	bytes(bits) per sensor value	max 16 sensors FRTPS payload in bytes	Quantization Levels (Q1/Q2/Q3) (Bits/value)
1	Low data rate (**Lo**)	1	4(32)	64	1xAgg.Level
2	Little data rate (**Li**)	2	4(32)	128	2xAgg.Level
3	Medium data rate (**M**)	3	4(32)	192	3xAgg.Level
4	High data rate (**H**)	4	4(32)	256	4xAgg.Level
5	very High data rate/low quantization (**L1**)	5	3(24)	240	5xAgg.Level/Q1 Level(24)
6	High data rate/low quantization (**L2**)	4	3(24)	192	4xAgg.Level/Q1 Level(24)
7	Medium data rate/low quantization (**L3**)	3	3(24)	144	3xAgg.Level/Q1 Level(24)
8	Little data rate/low quantization (**M1**)	2	3(24)	96	2xAgg.Level/Q1 Level(24)
9	Little data rate/medium quantization (**M2**)	2	2(16)	64	2xAgg.Level/Q2 Level(16)
10	Low data rate/low quantization (**M3**)	1	3(24)	48	1xAgg.Level/Q1 Level(24)
11	Low data rate/medium quantization (**H1**)	1	2(16)	32	1xAgg.Level/Q2 Level(16)
12	Low data rate/high quantization (**H2**)	1	1(8)	16	1xAgg.Level/Q3 Level(8)
13	Maintain data rate/Zero payload (**H3**)	1	0(0-Header only)	0	-

Packet aggregation transforms the FRTPS frame into a multi packet structure of compressed or non compressed values. This structure is similar to the FRTPS frame of Figure 1, with an aggregation upper limit of five aggregated packets per frame (Table 1, L1), or quantization with an upper limit of 8bit compressed 32bit sensory value for 1 aggregated packet per frame. The FRTPS protocol uses the *Agg. Id* field in its header so as to indicate the number of packets aggregated in each frame. The maximum FRTPS frame includes an aggregation of 5 packets/frame that in turn include a payload of 16 sensory 32bit values. The maximum FRTPS frame size is:$FRTPS_{p_{size}} = Eth_{header_{(14)}} + IP_{header_{(20)}} + UDP_{header_{(8)}} + FRTPS_{frame_h eader_{(8)}} + FRTPS_{payload_{(16)}} \cdot n_{packets} \cdot \frac{n_{bytes}}{frame} \leq 306bytes.$

Using the previous formula the minimum FRTPS packet size is the H2 SRL level (Table 1, H2 level) with 16x8 bit sensory values and 1 packet per frame. That is a frame of 66bytes, with a 50bytes of header overhead header. Also a H3 SRL level (Table 1, H3 level) exists in the FRTPS protocol for service probing purposes and this level frames carry no payload data. A FRTPS frame at SRL=1 level (Table 1, Lo level), must include at least more than 14 sensory values for the FRTPS payload to be a little bigger than its header. This is an excessive protocol overhead not caused by FRTPS itself but from the lower level medium, networking and transport protocols.

An FRTPS source initially starts transmitting data using the base FRTPS level ranked as Lo (Table 1, Lo -low aggregation and zero quantization level). If limited network resources, FRTPS first increases its aggregation level by moving from 1 to a maximum of 5 packets per frame (Table 1, FRTPS levels: little (Li), middle(M), high (H), very High(L1)). From L1 and thereafter, quantization levels occur and the transmitted payload values are compressed using uniform compression method similar to the α-law used at PCM A2D value sampling. In quantum level Q1, the number of bytes per sensory data is reduced to 3 bytes, for Q2 to 2 bytes and for Q1 up to a single byte. The sensory payload value compression mapping performed in quantization Q-levels follows Equation 1.

$$X'_{value} = \frac{X_{value}}{2^{Q_{i+1}} - 1} \tag{8}$$

where X_{value} is the current quantization level Q_i input value and Q_{i+1}, $(Q_{i+1} < Q_i)$ the next quantization level bits (24,16,8). Output value is the quantization process result value X'_{value}.

3.3 FRTPS Fuzzy control mechanism

The FRTPS protocol supporting mechanism is consisted of: 1. A flow monitoring agent, instantiated at the transmitter and receiver ends, 2. the Fuzzy controller module, implemented at the sender as part of its flow monitoring agent, 3. the data transmission client and data reception service and queues implemented at the sender and receiver accordingly. The functional parts of the FRTPS mechanism and fuzzy controller are illustrated at Figure 2.

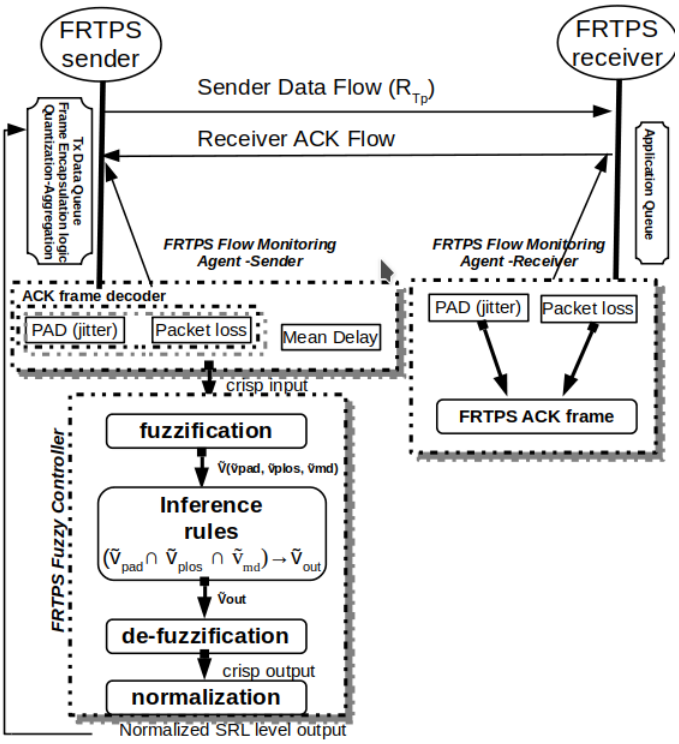


Figure 2: FRTPS flow monitoring mechanism and fuzzy controller

The fuzzy controller processes the crisp input metric values acquired by the transmitter monitoring agent as follows:

Step 1 : Fuzzification process, calculating the fuzzy input vector of PAD, Packet loss and Mean Delay V=(vpad, vplos, vmd) based on the applicable input degree of membership functions as illustrated at Figure 3. For the crisp values of: $X_{PAD} = 3, X_{PLOS} = 0.005, X_{MD} = 32$, the membership function values for the participating terms are: $\mu_L(PAD) = 0.25$, $\mu_G(PAD) = 0.5$, $\mu_L(PLOS) = 0.33$, $\mu_{Sp}(PLOS) = 0.66$ and $\mu_L(MD) = 0.4$, $\mu_G(MD) = 0.2$.

Step 2: Inference Pass. The fuzzy set values pass through a set of rules shown at Table 3. The inference logical conjuction operators are based on the minimum degree of membership value among the participating terms. The specified fuzzy terms threshold values and have been derived from previous experimentation on the AMESETP protocol [30].

The FRTPS logic lies on the aggregation of packets per frame for transient delays and random drops. For mild congestion incidents FRTPS follows more of a conservative policy, trying by reducing the number of packets per flow while quantizing the bytes per sensor value field, compressing the input values inside each packet. For severe congestion incidents, FRTPS reduces its frame packets aggregation to 1pkt/frame, while maximizing the payload values quantization.

This mapping is presented at Table 1, columns 3, 4, 5. The translation of the fuzzy operators joining propositions into T-norms are calculated by taking the minimum value of the terms. Considering the FRTP rules set, the degree of activation per each participating rule using the previous example degree of membership values per attribute is shown at Table 2. Based on the activation values of the inference output, The Takagi-Sugeno weight-based defuzzifier is used to calculate the crisp output [53]. A simplified defuzzifier logic has been selected instead of a more complicated Takagi-Sugeno-Kang, ANFIS one [33, 46], in order for the controller to perform fast SRL calculations in terms of processing time.

Table 2: FRTPS example of inference rules and activation value of the SRL output, for each participating set of MD, PAD, PLOS DOM values.

PAD	**PLOS**	**MD**	**SRL**	μ_{SRL}
$L_{0.25}$	$L_{0.33}$	$L_{0.4}$	Lo(1)	0.25
$L_{0.25}$	$L_{0.33}$	$G_{0.2}$	Lo(1)	0.2
$L_{0.25}$	$Sp_{0.66}$	$L_{0.4}$	Lo(1)	0.25
$L_{0.25}$	$Sp_{0.66}$	$G_{0.2}$	Li(2)	0.2

PAD	**PLOS**	**MD**	**SRL**	μ_{SRL}
$G_{0.5}$	$L_{0.33}$	$L_{0.4}$	Lo(1)	0.33
$G_{0.5}$	$L_{0.33}$	$G_{0.2}$	Lo(1)	0.2
$G_{0.5}$	$Sp_{0.66}$	$L_{0.4}$	Li(2)	0.4
$G_{0.5}$	$Sp_{0.66}$	$G_{0.2}$	Li(2)	0.2

Step 3: Defuzzification process. The output variable $\tilde{y} =< \widetilde{SRL} >$ has terms that equal to a constant value that equals to the SRL id, and for each term, its membership function corresponds to the FLC weight $w_i(j)$. The final chirp output y is calculated as a weighted average defuzzifier according to Equation 9

$$y_{out} = \frac{\sum_{j=1}^{k} w_i(j) \cdot y(i)_j}{\sum_{j=1}^{k} w_i(j)} \tag{9}$$

where $i = 1..13$ is the SRL membership function, inference output value and $w_i(j)$ are the corresponding rules outcome activation values as weights. Considering the output variable $\tilde{SRL}$, containing the output terms: Lo(1), Lo(1), Lo(1), Li(2), Lo(1), Lo(1), Li(2) and Li(2), as values 1,1,1,2,1,1,2,2, with weights equal to their degree of activation $\mu(SRL)$ (from Table 2), the crisp output equals to: $y = \frac{1 \cdot 0.25 + 1 \cdot 0.2 + 1 \cdot 0.25 + 2 \cdot 0.2 + 1 \cdot 0.33 + 1 \cdot 0.2 + 2 \cdot 0.4 + 2 \cdot 0.3}{0.25 + 0.2 + 0.25 + 0.2 + 0.33 + 0.2 + 0.4 + 0.2} = 1.41$.

Step 4: Smoothing. The crisp output smoothing process is performed according to equation 10

$$y'_{out} = 1 + ((1 - \alpha)\bar{y_{out}} + \alpha \; y_{out}) \tag{10}$$

where $\bar{y_{out}}$ is the mean calculated value of the k previous measurements of PAD, PLOS and MD and $\alpha > 0.5$ is a coefficient parameter. In our case $\alpha = 0.9$ and $k = 100$. Finally, the y'_{out} passes through a rounding step as follows, in order to determine the final SRL level:

$$SRL = floor((y'_{out}) mod(1) + 0.5), \; floor(x) = f(n, d) = \frac{n}{d} - \frac{n \cdot mod(d)}{d} \tag{11}$$

FRTPS fuzzy controller logic tries to identify to which SRL level the transmitter should adapt, as indicated by the receiver. Since fuzzy controllers are processing demanding, the fuzzy mechanism and rate adaptation are performed by separate threads. Furthermore, the FRTPS smoothing step the FRTPS rate adaptation mechanism smooth responsive with adaptive smoothing factor (α). FRTPS is more accurate in most cases than simple threshold-based solutions [8, 30, 34, 60]. FRTPS uses three state terms for the MD metric (low, good and high). Similarly to MD, FRTPS Fuzzy classifier uses four state terms (low, good, high, critical) for PAD metric and four state terms (low, spurious, transient, congested) for PLOS metric values, as depicted at Figure 3.

The crisp inputs fuzzification process is as follows: From estimated crisp metric values of MD, PLOS and PAD at the sender, the process initiates. The metric values correspond to membership functions as denoted from Figure 2 with a degree of membership (DOM) expressed in [0-1] scale. The fuzzy-processes used for PAD and Packet loss are triangle functions, while for MD trapezoid. That is,

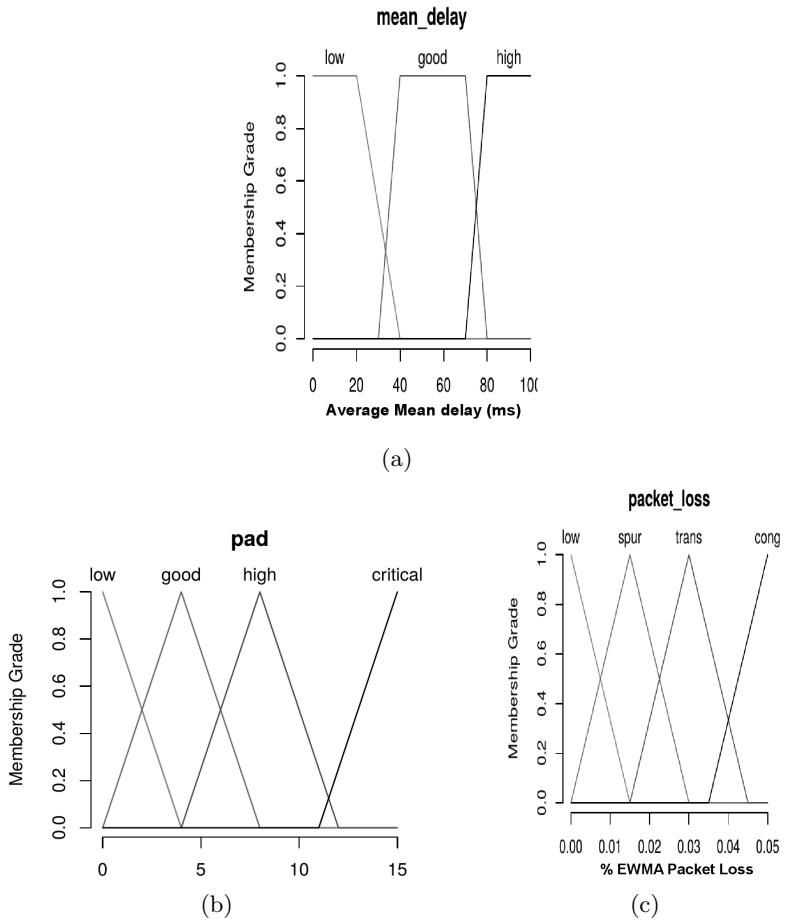


Figure 3: (a) FRTPS MF Fuzzy states for average Mean Delay (MD) - x axis in (ms) (b) FRTPS MS Fuzzy states for Packet Arrival Deviation (PAD) - x axis in (ms) (c) FRTPS MF Fuzzy states for Packet Loss (PLOSS) - x axis, % frame loss after EWMA process

the membership values can be easily calculated using linear interpolation. Afterwards, the inference process occurs. A set of conditional statements that determine the relation between input and output linguistic variable terms as shown at Table 3.

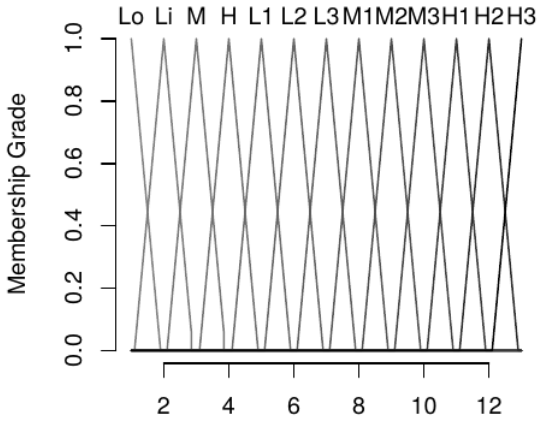


Figure 4: FRTPS Fuzzy output MF Degree Of Membership(DOM) for SRL rate-levels (1-13)

The MD, PLOS and PAD sets pass through the fuzzy inference rules providing the system with fuzzy output levels, each one with a different activation output value. The fuzzy controller rules OR and AND operators follow Zadeh's [14] T-norm (min) and S-norm (max) DOM operation accordingly

Table 3: FRTPS protocol fuzzy rules - SRL levels adaptation table ((a) MD=Low, (b) MD=Good and (c) MD=High)

(a) MD=Low - SRL levels

PAD \ PLOS	Low	Spurious	Transient	Congestion
Low	Low(Lo,1)	Low(Lo,1)	Little(Li,2)	Little(Li,2)
Good	Low(Lo,1)	Little(Li,2)	Little(Li,2)	Medium(M,3)
High	Little(Li,2)	Medium(M,3)	Medium(M,3)	Medium(M,3)
Critical	Medium(M,3)	High(H,4)	High(H,4)	High(H,4)

(b) MD=Good

PAD \ PLOS	Low	Spurious	Transient	Congestion
Low	Low(Lo,1)	Little(Li,2)	Little(Li,2)	Medium(M,3)
Good	Low(Lo,1)	Little(Li,2)	Medium(M,3)	High(H,4)
High	Medium(M,3)	Low-Q1(L1,5)	Low-Q2(L2,6)	Low-Q3(L3,7)
Critical	Low-Q2(L2,6)	Low-Q3(L3,7)	Medium-Q1(M1,8)	Medium-Q2(M2,9)

(c) MD=High

PAD \ PLOS	Low	Spurious	Transient	Congestion
Low	Medium(M,3)	Low-Q1(L1,5)	Low-Q2(L2,6)	Low-Q3(L3,7)
Good	High(H,4)	Medium-Q1(M1,8)	Medium-Q2(M2,9)	Medium-Q3(M3,10)
High	Low-Q1(L1,5)	Medium-Q3(M3,10)	High-Q1(H1,11)	High-Q2(H2,12)
Critical	Low-Q2(L2,6)	High-Q1(H1,11)	High-Q2(H2,12)	High-Q3(H3,13)

in order for the final SRL level set calculation. The output of the fuzzy controller is the SRL level selected for the next FRTPS time interval $T_{p_{int}}$. Figure 4 shows the MF DOM values for the entire 13 SRL output levels of FRTPS. As another testing FRTPS example, for MD=Low and for PLOS<0.01, moving the PAD metric value from 4ms to 13ms, shall provide the FRTPS system with an output value of SRL=1 for 4ms to SRL=4 for 13ms.

When the SRL level is calculated by the fuzzy controller the sender immediately alters its transmission frame rate or enforces payload quantization (if decided by the controller). Since the fuzzy controller exists in both sender and receiver ends, the receiver end has already calculated the new SRL level and expects the sender to adjust its frame rate to the new SRL level. The time interval needed for this SRL change is the *pre-adjust interval*, $T_{p_{int}}$ and is equal to: $\max(\frac{3}{2}\frac{\bar{RTT}}{2}, RTT_p)$, where $\frac{\bar{RTT}}{2}$ is the mean time for an ACK frame to reach the sender and RTT_p is the cumulative time ($\sum_{i=1}^{p}\frac{RTT_i}{2}$), for p consecutive ACK frames that reach the sender (set by the haptic application layer).

SRL crisp output values can also be in between two different neighborhood SRL levels, according to Figure 4. For this purpose a final modulo stage process takes places at the smoothing step. For example a crisp output of 2.3 corresponds to rate-level with SRL=2 (Li-little rate increase), while fuzzy output of 2.56 corresponds to rate level with SRL=3 (Medium rate increase).

4 FRTPS protocol performance evaluation

For the FRTPS preliminary testing implementation and design of the SRL levels, the R programming language has been used with the R fuzzy package for the implementation of FRTPS fuzzy controller [17]. The testing data set used was taken from the AMESETP dataset measurements [30], as well as UDP flow measurements using netem [22] and iPerf [24].

In order to evaluate the proposed FRTPS protocol, the authors compared the FRTPS protocol with the performance of RTP (UDP-based) protocol, most commonly used by real-time applications. After

preliminary tests, the authors implemented the FRTPS protocol in C++ using Qt and FuzzyLite [57]. For the RTP service implementation the JRTPLIB was used [48]. The authors constructed a real world scenario and sent streams of multi-sensory FRTPS and RTP data between two computers. The real-time testing dataset included movement commands of a 5-axis servo robotic arm, captured by a custom RPi v2 data-logger. The first computer referred to as the sender, is an Intel Core i3 8100 3.6Ghz with 4GB RAM. The other computer referred to as the receiver, is an Intel Core i7 3770 3.4Ghz with 8GB RAM Both computers are interconnected through the Greek Universities network (GRNET) with 100 Mbps up-link and down-link connectivity. The network experimental topology is illustrated at Figure 5.

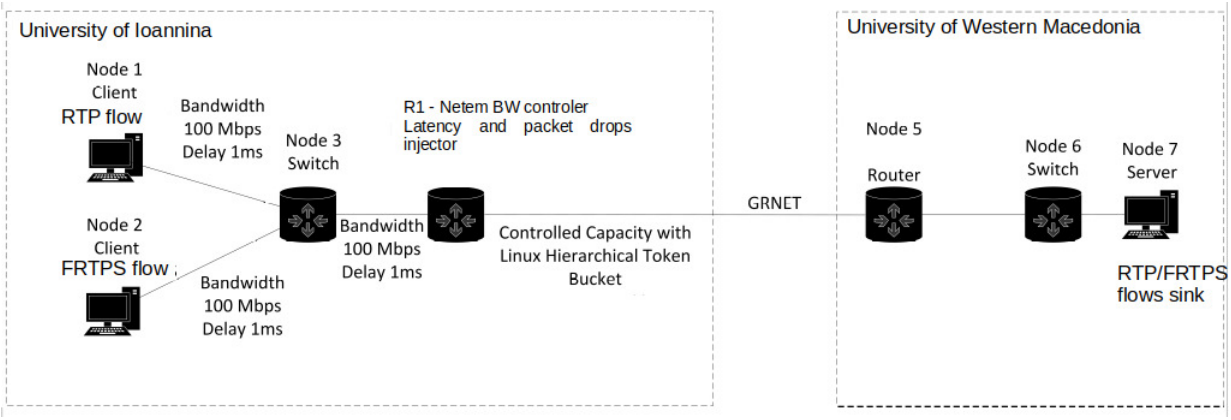


Figure 5: Network Topology of the performance evaluation scenario

In order to control the network conditions and provide saturated Internet links the authors used the network emulator netem [22, 27] and the Hierarchical Token Bucket (HTB) queuing discipline of the Linux Traffic Control utility (tc)[35] at the Linux router R1. two flows of 20000 packets are sent from each one of the clients (RTP flow and FRTPS flow) to the server. The payload of each packet is set to 64 bytes of payload that corresponds to 16 4byte sensory values. The assigned bandwidth of the network connection between client and server is arbitrary set to 700kbps and 1.4Mbit/s at the HTB queue. The sender rate for both RTP and FRTPS flows at the 700Kbps link is set to R=500frames/s and 1000frames/s for the 1.4Mbit/s link.

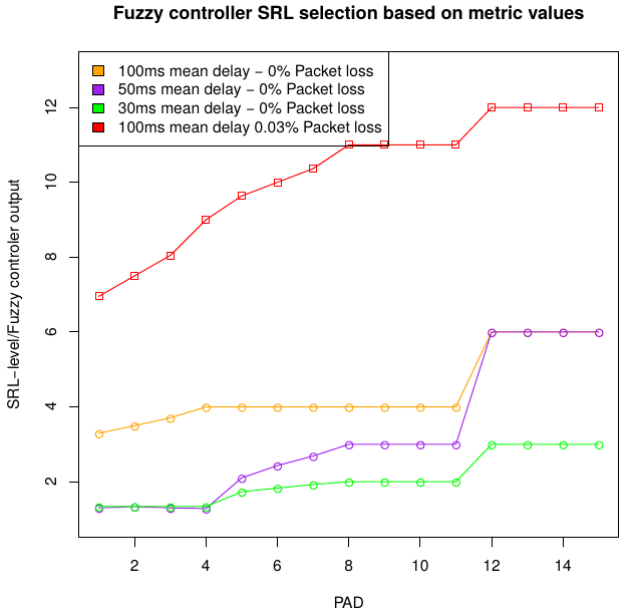
4.1 FRTPS fuzzy controller experimentation

In this experimental scenario the FRTPS controller is put to the test, by introducing different types of PAD variances to the FRTPS client and monitoring the crisp fuzzy output SRL level selected for each case. As illustrated at Figure 6a the client FRTPS controller is being tested by deliberate sending bogus ACKS from the receiver indicating different received frame deviations based on Equation 3 that in turn lead to different PAD value calculations by the sender and different crisp outputs by the controller.

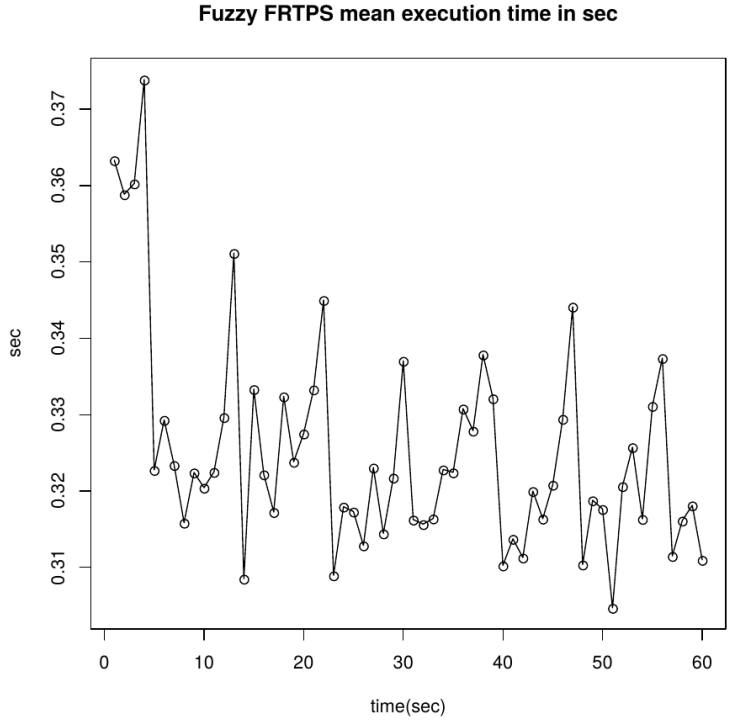
The FRTPS experimentation is performed by statically setting at the sender the average mean delay variable values to 30ms 50ms and 100ms, values that indicate mild and severe network contention prior to the appearance of router queue drops and the initiation of a burst congestive incident. It is probable that under such conditions random drops occur as well as excessive queuing delays. This is the reason for calculation of mean delay values 100-10000 times bigger than the flow transmission rates. The FRTPS SRL selection starts from SRL level=1(Lo) and ends up to the SRL level=6 (L2). Apart from SRL level 6, all previous levels are packet aggregation levels, where multiple packet values (payloads) are packed together into one frame. Moreover, in Figure 6a - red line, the authors also introduced random errors to the receiver ACK PLOS field, in order to confirm the fuzzy response of the FRTPS controller to the quantization levels (6-13). This denotes that the entrance of the sender to one of the quantization levels occurs only in very extreme contention cases that are followed by random frame drops or in congestive incidents detected at the receiver and propagated back to the

sender via the PLOS ACK field.

Finally, at Figure 6b the mean CPU processing time of the FRTPS controller over execution time is illustrated. It is obvious that the average processing time of an ACK, metric values re-calculation and prediction of the next SRL transmission level by the controller CPU processing costs to the authors' experimental client are: 310-370ms. That is, if the control prediction process has been performed for every received ACK the maximum achieved transmission rate T_{int} will be no more than 2-3 frames per second for a single core CPU. This is an indication that the fuzzy prediction calculation will be performed every n ACKs received by the sender ($T_{calc} \geq 1000T_{int}$) and that a complete CPU core has to be utilized for the monitoring of the fuzzy control process.



(a)



(b)

Figure 6: (a) FRTPS fuzzy states traversal over different estimated PAD values by the client - x axis in (ms) (b) FRTPS controller mean processing delay

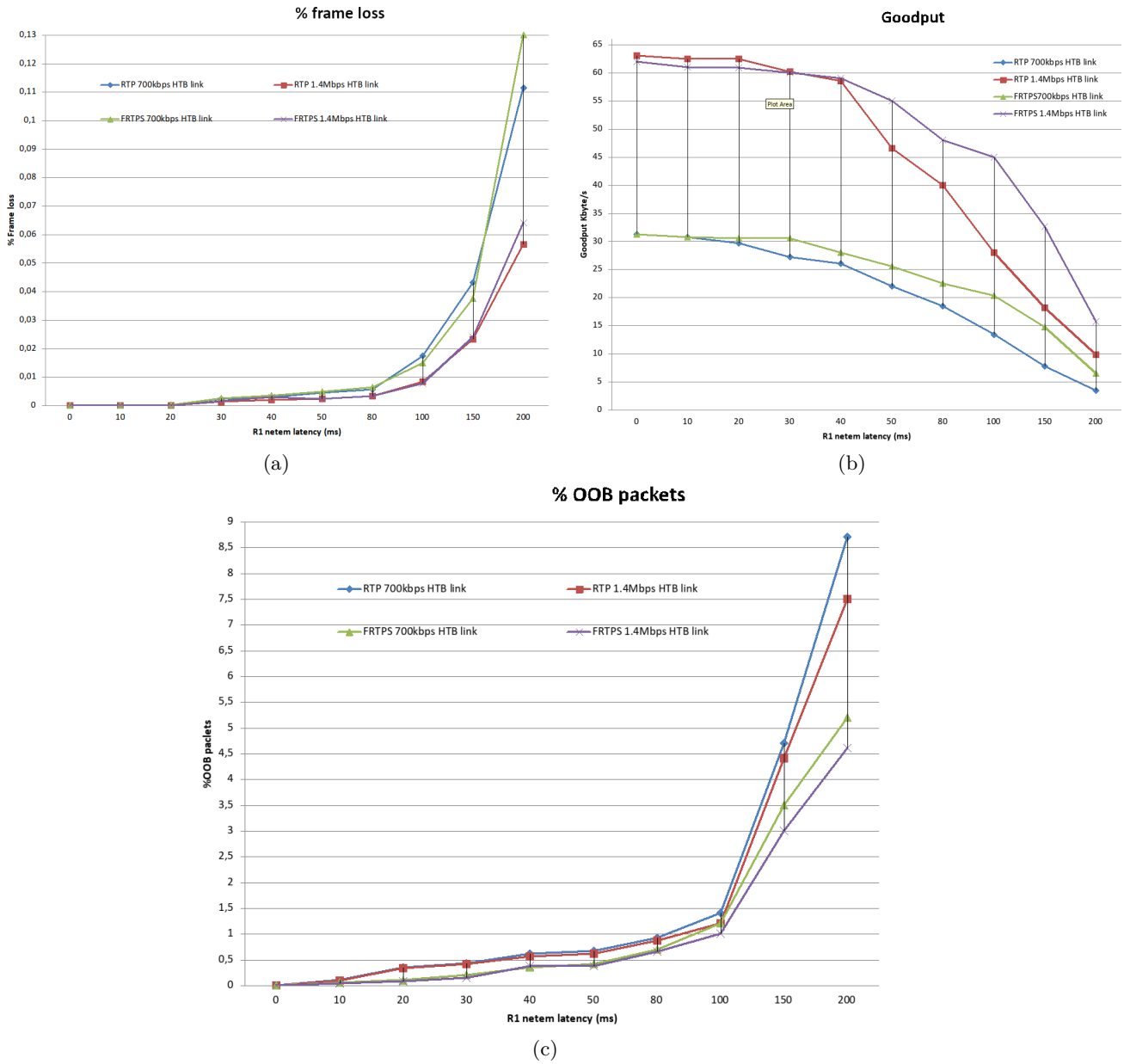


Figure 7: (a) %Frame Loss vs R1 Latency for 700kbps, 1,4Mbps channel capacity for the RTP and the FRTPS protocols. (b) Goodput Kbytes/s vs R1 Latency for 700kbps, 1,4Mbps channel capacity for the RTP and the FRTPS protocols. (c) % Packets out of Order vs R1 Latency for 700kbps, 1,4Mbps channel capacity for the RTP and the FRTPS protocols.

4.2 Experimentation on saturated links due to contention increase

In this scenario 0% packet loss has been injected to the netem queue. The queuing delays (latency) in the netem queues at the R1 are set to 0-50, 80, 100, 150 and 200msec and the netem queue jitter at the 90% of the corresponding latency value.

The metrics used in the experiments are the frame loss, goodput and the total number of packet that arrived out of order at the receiver end. The receiver average goodput is measured in KBytes/s of data payload/s at the receiver queue. Experimental results are shown at Figure 7.

Comparing the experimental results of the RTP and the FRTPS flows at Figure 7a, it is clear that in the examined scenario both RTP and FRTPS flows experience similar packet losses. This is due to the fact that the FRTPS operates on top of RTP too and copes with the same re-transmission mechanisms as those of RTP. It is like having two aggressive UDP flows competing over the same bottleneck. Nevertheless FRTP payload is far more different than that of RTP due to the fact that

it uses payload aggregation and quantization. Aggregation of frames is the prevailing policy up until 50-100ms of inserted latency while above 100ms both aggregation and quantization fuzzy states reside.

From Figure 7b and for latency up until 150ms the benefits of payload aggregation (packets aggregation) into single frames are obvious. For the bottleneck scenario of 700kbps, From 30-150ms queuing latency the average goodput of FRTPS is 21% more with respect to RTP. similar results apply for the bottleneck scenario of 1,4Mbit/s starting from 40ms up to 100ms with 25% more goodput delivered from FRTPS. For latency disciplines above 150ms and up to 200ms, where both quantization and aggregation apply, in the 700kbps scenario the FRTPS goodput is 89% more than RTP while for the FRTPS goodput for the 1,4Mbps scenario is 70% more than that delivered from the RTP flow. In total in a contenting environment among FRTPS and RTP, the FRTPS flows can deliver an average of at least 26% more data to its destination under the same network conditions, experiencing similar frame losses and frame re-transmission policies. That is, FRTPS protocol outperforms the RTP protocol in a contenting environment of limited resources.

Comparing RTP and FRTPS Out Of Band (OOB) packets at Figure 7c for the 700Kbps bottleneck scenario, the FRTPS flow delivers in average 0,75% less OOB packets than RTP. The same applies for the 1,4Mbps/s bottleneck where the FRTPS delivers 2% less OOB messages. The 7,9%-10% correspond to 150-400 less Out Of Band (OOB) packets of a total of 20000 frames.

5 Conclusions

The authors propose a new fuzzy protocol called FRTPS that maintains rate adaptive and data quantization mechanisms. The authors present the proposed protocol and put to the test its implementation functionality. Functionality test of the FRTPS controller have shown that the FRTPS controller operates well distinguishing contention incidents where a payload aggregation policy is most applicable, from random drop incidents where both aggregation and quantization is the best policy to offer, from congestive incidents where quantization and discrete payload reduction is the most network friendly option. The proposed FRTPS algorithm adaptive decisions are based on three different crisp metric inputs, the mean delay, the PAD and the packet loss, as calculated by the receiver end and propagated to the sender using the FRTPS protocol. The fuzzification process is implemented at the transmitter end, handed out by the receiver end measurements.

The authors experimented with the FRTPS performance on a real-case scenario and compared the performance of the FRTPS with that of the RTP protocol. From the experimental results it is obvious that under similar network conditions where the two flows compete for the same limited resources the FRTPS protocol outperforms the RTP protocol, delivering 26% more payload data to its destination. Regarding data synchronization, it is pinpointed that FRTPS can transfer at least 1% more synchronized frames than RTP, due to its adaptive packet aggregation mechanism. It is set for further experimentation how the payload quantization affects synchronized data deliveries.

Finally, measurements of the processing delays enforced by the FRTPS protocol to the sender. From the experimental results, the processing time requirements of fuzzy algorithms are at least 30-300 times more than the processing time requirements of the RTP protocol. However, since real time flow interruption is considered to be dangerous and problematic in most application cases, The authors set as a future work the implementation of custom ASIC cores with embedded schedulers that will optimize receiver end captured measurements and reduce significantly fuzzification processing time overheads.

Acknowledgements

The authors would like to thank the Dept. of Business Administration, University of Western Macedonia, Greece, for his participation in the experimental setup, server provision and provided network that made the execution of the experimental scenarios possible.

References

[1] Al-Saadi, R.; Armitage, G.; But, J.; Branch, P. (2019). A Survey of Delay-Based and Hybrid TCP Congestion Control Algorithms, *IEEE Communications Surveys Tutorials*, 21(4), 3609–3638, 2019.

[2] Akács, Á.; Kovács, L.; Rudas, I. J.; Precup, R. E.; Haidegger, T. (2015). Models for force control in telesurgical robot systems, *Acta Polytechnica Hungarica*, 12(8), 95–114, ISSN: 1785-8860, 2015.

[3] Antonakoglou, K.; Xu, X.; Steinbach, E.; Mahmoodi, T.; Dohler, M. (2018). Toward Haptic Communications Over the 5G Tactile Internet, *IEEE Communications*, 20(4), 3034–3059, 2018.

[4] Attiya, G. (2012). New Strategy for Congestion Control based on Dynamic Adjustment of Congestion Window, *International Journal of Computer Science Issues*, 9(2), 368–377, 2012.

[5] Awang N. S.; Alubady, R.; Abduladeem, K. W. (2017). Simulated performance of TCP, SCTP, DCCP and UDP protocols over 4G network, *The 8th International Conference on Advances in Information Technology*, 111(2017), 2–7, 2017.

[6] Benítez-Pérez, H.; Ortega-Arjona, J.; Esquivel-Flores, O.; Rojas-Vargas, J. A.; Álvarez-Cid, A. (2016). A Fuzzy Networked Control System Following Frequency Transmission Strategy, *International Journal of Computers Communications & Control*, 11(1), 11–25, 2016.

[7] Bermejo, C.; Hui, P. (2017). A survey on haptic technologies for mobile augmented reality, *arXiv:1709.00698 [cs]*

[8] Boukerche, A.; Maamar, H.; Hossain, A. (2008). An efficient hybrid multicast transport protocol for collaborative virtual environment with networked haptic, *Multimedia Systems*, 13(4), 78-83, 2008.

[9] Chatterjee, A.; Chatterjee, R.; Matsuno, F.; Endo, T. (2008). Augmented Stable Fuzzy Control for Flexible Robotic Arm Using LMI Approach and Neuro-Fuzzy State Space Modeling, *IEEE Transactions on Industrial Electronics*, 55(3), 1256–1270, 2008.

[10] Chowdhury, I.S.; Lahiry, J.; Hasan, S. F. (2009). Performance analysis of Datagram Congestion Control Protocol (DCCP), *12th International Conference on Computers and Information Technology*, 454–459, 2009.

[11] Clark, A.; Qu, Q. (2012). RTCP XR Packet Delay Variation, [Online]. Available: https://ietf.org/rfc/rfc6798.html.

[12] Dapeng, W.; Hou, Y.T.; Wenwu, Z.; Ya-Qin, Z.; Peha, J.M. (2001). Streaming video over the Internet: approaches and directions, *IEEE Transactions on Circuits and Systems for Video Technology*, 11(3), 282–300, 2001.

[13] Dodeller, S.; Georganas, N. D. (2004) Transport Layer Protocols for Telehaptics Update Messages, *Proceedings of the 22nd Biennial Symposium on Communications*, 1(1), 112–118, 2004.

[14] Dzitac, I.; Filip, F.G.; Manolescu, M.J. (2017). Fuzzy Logic Is Not Fuzzy: World-renowned Computer Scientist Lotfi A. Zadeh, *International Journal of Computers Communications & Control*, 12(6), 748–789, 2017.

[15] Even, R.; Xia, F. (2012). RTP Control Protocol (RTCP) Extension for a Third-Party Loss Report, [Online]. Available: https://tools.ietf.org/html/rfc6642. Accessed on: Mar. 2014

[16] Floyd, S.; Handley, M.; Kohler, E. (2006). Datagram Congestion Control Protocol (DCCP), [Online]. Available: https://tools.ietf.org/html/rfc4340, 2006.

[17] Gagolewski, M.; Caha, J. (2012). A Guide to the FuzzyNumbers Package for R, [Online]. Available: https://cran.r-project.org/web/packages/FuzzyNumbers/vignettes/FuzzyNumbersTutorial.pdf. Accessed on: Feb. 2016.

[18] Ghassan, A. A.; Mahamod, I.; Kasmiran, J. (2011). A Survey on Performance of Congestion Control Mechanisms for Standard TCP Versions, *Australian Journal of Basic and Applied Sciences*, 5(12), 1345–1352, ISSN: 1991-8178, 2011.

[19] Gil, R.P.A.; Johanyak, Z.C.; Kovacs, P. (2018). Surrogate model based optimization of traffic lights cycles and green period ratios using microscopic simulation and fuzzy rule interpolation, *International Journal of Artificial Intelligence*, 16(1), 20–40, 2018.

[20] Gupta, R.; Tanwar, S.; Tyagi, S.; Kumar, N. (2019). Tactile-Internet-Based Telesurgery System for Healthcare 4.0: An Architecture, Research Challenges, and Future Directions, *IEEE Network*, 33(6), 22–29, 2019.

[21] Ha, J.S.; Kim, S.-T.; Koh, S. J. (2005). Performance Comparison of SCTP and TCP over Linux Platform, *Advances in Intelligent Computing*, 396–404, 2005.

[22] Hemminger, S. (2005). Network emulation with Netem, [Online]. Available: http://developer.osdl.org/shemminger/netem, Accessed on Jun. 2016.

[23] Ickin, S.; Wac, K.; Fiedler, M.; Janowski, L.; Hong, J.-H.; Dey, A. K. (2012). Factors influencing quality of experience of commonly used mobile applications, *IEEE Communications Magazine*, 50(4), 48–56, 2012.

[24] iPerf, [Online]. Available: https://iperf.fr/iperf-doc.php, Accessed on May 2016.

[25] Jacobson, V. (1988). Congestion avoidance and control, *Symposium proceedings on Communications architectures and protocols*, 314–329, 1988.

[26] Jacobson, V. ; Frederick, R. ; Casner, S. ; Schulzrinne, H. (2003). RTP: A Transport Protocol for Real-Time Applications, [Online]. Available: https://tools.ietf.org/html/rfc3550.

[27] Jurgelionis, A.; Laulajainen, J-P.; Hirvonen, M.; Wang, A. I. (2011). An Empirical Study of NetEm Network Emulation Functionalities, *Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN)*, 1–6, 2011.

[28] Kawazoe, A.; Luca, M. D.; Visell, Y. (2019). Tactile Echoes: A Wearable System for Tactile Augmentation of Objects, *IEEE World Haptics Conference (WHC)*, 359–364, 2019.

[29] Khalid, M.; Said, H.; Asad, A.; Abubakr, E.S. (2018). Studying the TCP Flow and Congestion Control Mechanisms Impact on Internet Environment, *IJCSIS*, 16(11), 174–179, 2018.

[30] Kokkonis, G.; Kontogiannis, S.; Tomtsis, D. (2016). An open source architecture of a wireless body area network in a medical environment, *International Journal of Digital Information and Wireless Communications*, 6(2), 63–78, 2016.

[31] Kokkonis, G.; Psannis, K.; Roumeliotis, M.; Kontogiannis, S. (2012). A Survey of Transport Protocols for Haptic Applications, *16th Panhellenic Conference on Informatics*, 192–197, 2012.

[32] Kovács, L. ; Haidegger, T. ; Rudas, I. (2013). Surgery from a distance—Application of intelligent control for telemedicine, *IEEE 11th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, 125–129, 2013.

[33] Kukolj, D. (2002). Design of adaptive Takagi–Sugeno–Kang fuzzy models, *Applied Soft Computing*, 2(2), 89–103, 2002.

[34] Li, P.; Lu, W.; Sun, Z. (2005). Transport layer protocol reconfiguration for network-based robot control system, 1049–1053, 2005.

[35] Linux Advanced Routing & Traffic Control HOWTO, [Online]. Available: https://lartc.org/, Accessed on Sep. 2011.

[36] Madhuri, D.; Reddy, P.C. (2016). Performance comparison of TCP, UDP and SCTP in a wired network, *International Conference on Communication and Electronics Systems (ICCES)*, 1–6, 2016.

[37] Masse, R. A. C.; Ochoa-Zezzatti, A.; García, V.; Mejía, J.; Gonzalez, S. (2019). Application of IoT with haptics interface in the smart manufacturing industry, *International Journal of Combinatorial Optimization Problems and Informatics*, 10(2), 17–25, 2019.

[38] Mauve, M.; Hilt, V.; Kuhmunch, C.; Effelsberg, W. (2001). RTP/I-toward a common application level protocol for distributed interactive media, *IEEE Transactions on Multimedia*, 3(1), 151-161, 2001.

[39] Molia, H. K.; Kothari, Amit D. (2019). Fuzzy Logic Systems for Transmission Control Protocol, *Proceedings of the 2nd International Conference on Communication, Devices and Computing*, 553-565, 2019.

[40] Morton, A.; Claise, B. (2009). Packet Delay Variation Applicability Statement, [Online]. Available: https://tools.ietf.org/html/rfc5481.

[41] Nagy, T. D.; Haidegger, T. (2019). A DVRK-based Framework for Surgical Subtask Automation, *Acta Polytechnica Hungarica*, 16(8), 61–78, 2019.

[42] Osman, H. A.; Eid, M.; Saddik, A. E. (2008). Evaluating ALPHAN: A Communication Protocol for Haptic Interaction, *Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, 361–366, 2008.

[43] OTT: Beyond Entertainment Consumer Survey Report (2017). [Online]. Available: https://www.conviva.com/research/ott-beyond-entertainment/. Accessed on: Sep. 2019.

[44] Paulo R. (2013). Accenture video-over-internet-consumer-survey-2013, [Online]. Available: https://www.slideshare.net/ratinecas/accenture-videooverinternetconsumersurvey2013. Accessed on: May 2018.

[45] Phung, M. D.; Van Thi Nguyen, T.; Quach, C. H.; Tran, Quang Vinh (2010). Development of a tele-guidance system with fuzzy-based secondary controller, *11th International Conference on Control Automation Robotics Vision*, 2010.

[46] Precup, R. E.; Tomescu, M. L.; Dragos, C. A. (2014). Stabilization of Rössler chaotic dynamical system using fuzzy logic control algorithm, *International Journal of General Systems*, 43(5), 413–433, 2014.

[47] Reiter, U.; Brunnström, K.; De Moor, K. et al. (2014). Factors Influencing Quality of Experience *Quality of Experience: Advanced Concepts, Applications and Methods* 55–72, 2014.

[48] RTPLIB, [Online]. Available: https://research.edm.uhasselt.be/jori/page/CS/Jrtplib.html, Accessed Oct. 2018

[49] Shirmohammadi, S.; Georganas, N.D. (2000). Collaborating in 3D virtual environments: a synchronous architecture, *Proceedings IEEE 9th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE 2000)*, 35–42, 2000.

[50] Skurowski, P.; Gruca, A. (2008). SMPDV - A new jitter estimator proposal, *Studia Informatica*, 29(81), 1–16, 2008.

[51] Tabash, I.; Beg, M.S.; Ahmad, N. (2013). Improving TCP Performance based on Fuzzy-Logic for Mobile Ad hoc Networks, *International Journal of Computer Applications*, 78(2), 11–16, 2013.

[52] Takács, Á.; Rudas, I.; Bösl, D.; Haidegger, T. (2018). Highly Automated Vehicles and Self-Driving Cars [Industry Tutorial, *IEEE Robotics Automation Magazine*, 25(4), 106-112, 2018.

[53] Takagi, T.; Sugeno, M. (1993). Fuzzy Identification of Systems and Its Applications to Modeling and Control, *Readings in Fuzzy Sets for Intelligent Systems*, 387–403, 1993.

[54] Taylor, T.; Schwarzbauer, H. J.; Kalla, M. et al. (2000). Stream Control Transmission Protocol, [Online]. Available: https://tools.ietf.org/html/rfc2960.

[55] Trang, S. Q. V.; Lochin, E. (2016). FLOWER, an innovative Fuzzy Lower-than-Best-Effort transport protocol, *Computer Networks*, 110, 18–30, 2016.

[56] Uchimura, Y.; Ohnishi, K.; Yakoh, T. (2002). Bilateral robot system on the real time network structure, *7th International Workshop on Advanced Motion Control. Proceedings*, 51(5), 63–68, 2002.

[57] Vilela R. J. (2016). The FuzzyLite Libraries for Fuzzy Logic Control, [Online]. Available: https://www.fuzzylite.com/. Accessed on: Sep. 2016.

[58] Wang, D.; Ohnishi, K.; Xu, W. (2020). Multimodal Haptic Display for Virtual Reality: A Survey, *IEEE Transactions on Industrial Electronics*, 67(1), 610–623, 2020

[59] Wirz, R.; Ferre, M.; Marín, R.; Barrio, J. et al. (2008). Efficient Transport Protocol for Networked Haptics Applications, Haptics: Perception, Devices and Scenarios, Springer, 3–12, 2008.

[60] Wirz, R.; Marin, R.; Ferre, M. et al. (2009). Bidirectional Transport Protocol for Teleoperated Robots, *IEEE Transactions on Industrial Electronics*, 56(9), 3772–3781, 2009.

This journal is a member of, and subscribes to the principles of,
the Committee on Publication Ethics (COPE).
https://publicationethics.org/members/international-journal-computers-communications-and-control

Cite this paper as:

Kontogiannis, S.; Kokkonis, G. (2020). Proposed Fuzzy Real-Time HaPticS Protocol Carrying Haptic Data and Multisensory Streams, *International Journal of Computers Communications & Control*, 15(4), 3842, 2020. https://doi.org/10.15837/ijccc.2020.4.3842