User-Intent Aware Transport-Layer Intelligence for Frame Synchronisation in Multi-Party XR Application

Vu San Ha Huynh†*, Peng Qian†*, Ning Wang*, Carl Udora*, Rahim Tafazolli*

*5GIC & 6GIC, Institute for Communication Systems (ICS), University of Surrey, Guildford, Surrey, U.K.
Email: {v.sanha, peng.qian, n.wang, cu00029, r.tafazolli}@surrey.ac.uk

***Abstract*— Emerging immersive media applications demand tailored performance to accommodate diverse user intents, particularly in scenarios with multiple users with different intents and requiring frame synchronisation. This paper introduces a novel transport-layer intelligence scheme that leverages a user intent-aware API. This API enables the application layer to communicate specific user intents and requirements to the transport layer, optimizing immersive application performance. Using deep reinforcement learning, our solution automatically selects the optimal transport protocol and configuration for each user intent across various immersive scenarios. Our evaluation focuses on a live immersive video streaming application, with different users transmitting volumetric content under different network conditions. Results demonstrate that our scheme accurately identifies suitable transport protocols and tailored configurations for a wide range of user intents, ensuring multi-user frame Synchronisation.**

***Index Terms*— volumetric streaming, Transport-layer intelligence, Intent-based networking**

I. INTRODUCTION

In recent years, with the rise of interactive virtual reality devices and the emergence of various multi-party applications, how to effectively transmit live volumetric content over networks has become a hot topic in both the industry and academia [1-3]. However, only minor attention has been paid to transport layer protocols, which are responsible for probing network conditions and directly determining the transmission rate of the volumetric frame. The legacy transport layer protocols (e.g., SCTP, TCP, and QUIC [4-5]) and their congestion control algorithms (BBR, CUBIC [6-7]) lack the flexibility and programmability to meet the diverse needs of modern media applications, particularly those with varying user intents in live volumetric streaming scenarios. For instance, considering an Internet-scale live virtual performance, the members of a band can be located in different countries or cities, and perform corresponding musical instruments by receiving the volumetric images of other members. At the same time, after the volumetric content of all band members is transmitted to the audience in different regions, the audience can also transmit their interactive behaviours to the band in the form of volumetric content for real-time interaction. However, due to variations in personnel numbers, behavioural patterns, and the current stage's focus, the volume of transmitted content may significantly differ [1, 8]. In addition, the transmission of such multi-party content in different regions needs to undergo various network conditions. Therefore, the transport layer protocol is required to dynamically perceive the unpredictable user intent changes in the application layer and make real-time rate adjustments based on the changes in both the application layer and the network conditions. In other words, flexibility and extensibility are crucial, especially for multi-user Synchronisation, precluding a one-size-fits-all approach.

In this paper, we propose a transport-layer architecture that incorporates user intent awareness for immersive volumetric environments. This architecture exposes an intent interface at the application layer to communicate and indicate the user's specific requirements to the transport layer. User intent input can be translated into instant application performance requirements, and then the transport layer protocol will automatically conduct self-adaptive configuration to guarantee such requirements (e.g., multi-party frame Synchronisation). By doing so, we aim to enhance the flexibility and programmability of the transport layer, with minimal modifications to legacy applications while ensuring zero risk to the kernel. This architecture also eliminates the need for manual tuning of transport-layer parameters or expert knowledge of different transport stacks, enabling scalable and adaptive Synchronisation tailored to user intents. We employ deep neural networks enhanced by reinforcement learning for robust function approximation, capturing intricate representations of application use case intents and requirements. By leveraging and extending the actor-critic reinforcement learning framework [9], our approach enables the actor network to govern transport-layer configuration decisions, while the critic network assesses and provides feedback on the chosen configurations, tailored to specific user intents within immersive applications.

The main contribution of our work is listed as follows:

- We propose a holistic solution for user intent awareness at the transport layer in live volumetric immersive applications. Our approach includes an application-intent expression Application Programming Interface (API) and an intent policy manager to translate such user intent into detailed performance metrics. This allows the transport layer to adaptively support tailored application requirements across diverse use cases.
- We implement a Deep Reinforcement Learning (DRL)--based transport-layer scheme that identifies the most suitable protocols and configurations to meet user intent-specific performance requirements in live volumetric applications. This ensures the transport layer's extensibility and flexibility, enabling transparent utilization of existing protocols with intent-optimized configurations.
- In our evaluation, we compare our intent-aware transport-layer services against traditional approaches in a live volumetric streaming application, emphasizing multi-user

Synchronisation. Our system significantly improves performance in key metrics, predicting appropriate protocols and configurations to achieve desired application performance while balancing trade-offs.

The remainder of the paper is organized as follows: Section II reviews related literature, Section III details the framework architecture and DRL algorithm, Section IV analyses performance, and Section V concludes the paper.

II. Literature Review

Intent-driven networks (IDN) [10] is a recent approach that allows expressing service needs ("intents") through declarative or imperative mechanisms, abstracting away the complexity of their implementation. These intents, with the right level of abstraction, can comprehensively describe services or applications and their requirements, which are then communicated to lower network layers. IDN starts with a semantic language to represent intents, converting them to primitives and mapping them to executable policies. These policies are verified and deployed to the network to fulfil the original service intents. However, a unified and clear definition of IDN is still lacking, and its enabling techniques are under further exploration.

In recent years, various intent-aware and protocol-independent transport layers have been introduced, including IETF TAPS [11], NEAT [12], Socket Intents [13], and Congestion Control Plane (CCP) [14], aiming to enhance the flexibility and extensibility of the transport layer. IETF TAPS [11], a recent standard body effort, strives to replace the conventional system-level socket API with a new transport-layer socket API, enabling applications to articulate their needs and preferences for optimal transport-layer service selection. CCP [14], another recent advancement, modifies parameters like congestion window and sending rate in user space to customize the congestion control of underlying TCP implementations. Hybrid Information-Centric Network (ICN) Transport [15] leverages the ICN architecture, using prefixes in content naming to communicate application intents, such as real-time audio or video streams, and tags like "wireless," "cellular," "interactive," or "reliable" to indicate preferences. These tags are not static but can be dynamically updated. Researchers in [16] explored deep reinforcement learning to adjust TCP's congestion window, ensuring applications achieve their desired delays in dynamically changing networks. Leveraging advancements in network programmability and virtualization, numerous other cross-layer application intent awareness efforts have been proposed, including DiffServ [17], TMForum APIs [18], ETSI NFV Network Service Descriptor [19], and ETSI Mobile Edge Application Descriptor [20]. These efforts span data paths, user and kernel spaces, stream-byte and message-based approaches, demonstrating the ongoing innovation in intent-driven networking.

Although significant progress has been made in the next-generation intent-aware transport layer, available requests or API calls are still constrained by static APIs. CCP [14], AI-based congestion control [16], and transport layer adaptation [27] often rely on a specific transport protocol (e.g., TCP or QUIC) or predefined application requirements (e.g., throughput, delay), limiting their generality and adaptability. Research on IDN [10], emphasizing intent taxonomy and lifecycle functions, and Hybrid-ICN Transport [15] are ongoing but lack unified standards, implementations, verifications, and transport orchestration support. This underscores the need for further advancements in transport layer architecture to fulfil the demands of future user intent-aware networks.

III. Intent-Aware Transport-Layer Services

In this section, we describe the design of our intent-aware transport-layer architecture tailored for volumetric applications in which end users may express different intents on the applications in specific use case scenarios.

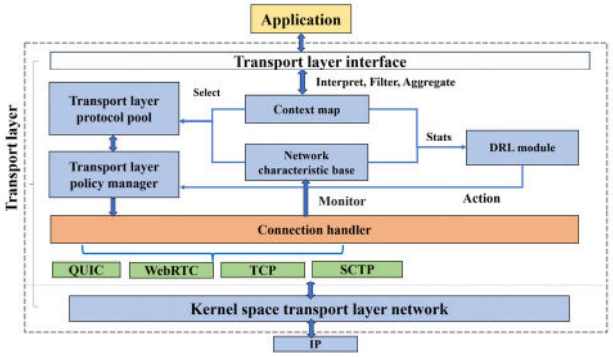

Figure 1. Intent-aware transport-layer services architecture for volumetric streaming environments

A. System Overview

We propose intent-aware transport-layer services that allow the extensible and flexible use of any transport-layer protocols with minimal modification at the application's source code (see Fig. 1). The connection handler is the key component to receive the configuration policy translated from user intent. By capturing the packets at the socket layer and redirecting them from the traditional data path, this connection handler allows to autonomously configure different connection properties (e.g. congestion control algorithm, congestion window (cwnd), initial window (IW), etc.) in response to different or changing of user intents in human-oriented immersive applications. Meanwhile, useful information from captured packets will be stored as input statistical data for a DRL module to train and generate an optimal configuration policy. These real-time configuration policies tailored for different user intents will be loaded by the transport layer policy manager and then instructed to the underlying protocol stack accordingly through the connection handler. Since the connection handler manages pre-activated sockets of different transport layer protocols (e.g., TCP, QUIC, SCTP), it will first select one of the protocols, and then configure its parameters. Moreover, the selected transport-layer configuration towards which the data path redirection is executed may either already exist in the Operating System (OS) or not, therefore in order to maintain reliability, our intent-aware transport-layer architecture integrates a systematic fallback to the application's original transport-layer protocol in any failure situation (middleboxes interferences, etc.). Consequently, this architecture is not limited to any particular

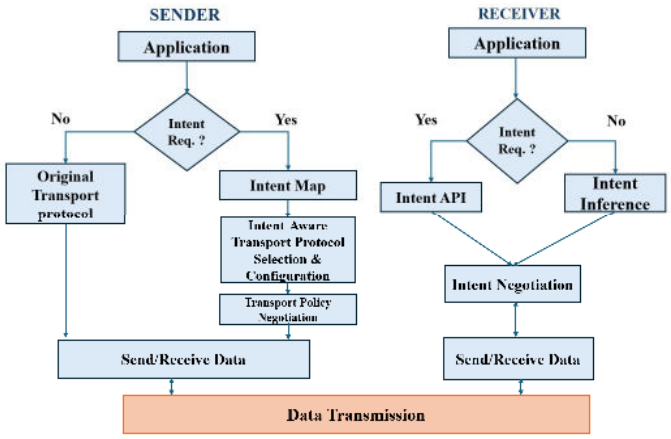

Figure 2. Simplified Workflow of the intent framework

implementation of the main execution loop which could be using eBPF [21] (for Linux kernel) or a customized datapath library (e.g. libccp [14]).

Fig. 2 shows an overview of intent-aware transport-layer services at the sender and receiver sides. Application performance targets or requirements in a specific intent are receiver-driven and can be flexibly expressed via our designed transport-layer API both at the beginning or in the middle of the data transmission session. The sender acquires frame request packets from the receiver and updates its intent map to keep track of applications with its corresponding user intent. Also, it performs a transport-layer protocol selection, and configuration optimization process with an aim to satisfy the requests of receivers. Application at the receiver side is allowed to express its requirement through our transport-layer API which updates the intent map at run time. The intent map allows us to map applications with their intent-dependent requirements as well as with its most suitable transport-layer protocols to satisfy these targets at a low computational cost. Packets received from the network card will be redirected to eXpress Data Path (XDP) [21] to perform high-speed packet processing before mapping back to the corresponding application. These packets received are also used for network profiling for future analysis.

In terms of detailed function implementation, at both the sender and receiver sides, the system interrupts the data path of the application's original transport-layer protocol (e.g. TCP) and takes control of the packets for our intelligent processing before the network stacks. The intent-aware transport-layer services capture system calls and network events (e.g. *sendmsg(), recvmsg()*) and then run our own safety-verified data-path programs at both user and kernel space. Assuming a host running the Linux operating system, the implementation using eBPF [21] hookers attached to root cgroupv2 [21] will enable every incoming and outcoming packet of all processes on the host to be captured and processed. With the assistance of a locally maintained intent/socket map, the socket and message controller are able to identify the specific socket and process to which the data packets should be forwarded. This map is updated in the socket and message controller every time a connection is established or closed.

B. Deep Reinforcement Learning-Based Intent-Aware Transport-Layer Protocol Selection & Configuration

Based on the filtered packet information, we utilize a deep reinforcement learning approach [9, 22, 25] in our integrated transport policy manager to automatically identify the most suitable set of transport-layer protocols and configurations to satisfy the user intent. We manage to investigate whether reinforcement learning "trial-and-error" approaches could be used to enable transport-layer intelligence where our considered scenarios have non-guaranteed global knowledge and a certain degree of dynamics in application intent-dependent requirements.

Input sample: The inputs into the transport policy manager comprise network characteristics profiling from the receiving packets (e.g., normalized round trip delay, packet loss). Based on that, the transport policy manager builds a table of network state and state transitions. The observed Quality-of-Service (QoS) metrics are utilized to infer the objective application-specific performance to which we compare the expected performance in the reward function. As the intent-aware transport policy manager maintains a historical record of state-action-reward tuples $< s_i, a_i, r_i >$, we describe our novel design of state and action spaces, and the reward function of the agent as follows:

State space: state s_i^t of endpoint n_i at time t is $s_i^t = \{u_i^t\}$ where $u_i^t = \{u_{i,0}^t, u_{i,2}^t, \ldots, u_{i,k}^t\}$ as the QoS and application-specific performance utility values observed. These QoS and application-specific performance utility values are bounded histories of statistics from received packet acknowledgement, and also the application layer feedback if it can be retrieved from the user intent interface. We propose to avoid metrics that are expected to be highly variable across connections just because of variations in link properties (e.g. deteriorated wireless signals).

Action space: action a_i^t of endpoint n_i at time t is defined as: $a_i^t = \{p_i^t, \forall p \in P\}$, $p_i^t = [0,1]$ indicating a list of transport-layer candidates associated with different configurations (e.g. Congestion Control (CC), cwnd, etc.). Our system model encourages exploration and avoids repeatedly selecting a particular set of transport parameters by assigning equal probabilities to actions having relatively the same Q-values. We manage to explore the action space where the transport policy manager selects the most suitable transport-layer protocols (e.g. TCP, UDP, QUIC, SCTP, etc.) and its transport-layer features (e.g. congestion control algorithms, cwnd, IW) based on the observed states to maximize the reward function. Our intent-aware transport-layer policy manager treats continuous cwnd configuration and discrete CC configuration separately. It is due to the huge space of the continuous action (i.e., the value of cwnd) that the DRL-agent can do at different times. Thus, in order to improve the feasibility and efficiency, we let the DRL agent find the best values of the action based on the parameters calculated by the underlying transport protocol (e.g. TCP). Thus, we propose a function that relates cwnd_{drl} value to the value of cwnd that the DRL-agent receives periodically from the state block:

$$\text{cwnd}_{\text{drl}} = c^{\alpha} \times \text{cwnd} \quad (1)$$

where c is a constant factor and $-1 \leq \alpha \leq 1$. Instead of searching the entire space, this simplifies the exploration phase and improves the learning convergence.

Reward space:
$r_i^t = -(req\_utility_i^t - curr\_utility_i^t)$ reflecting the gap between the intent-dependent request and the current experience. The reward model aims to reflect application-specific performance, possibly under QoS and other constraints, and the policy design that maximizes long-term rewards. As the reward may depend on the requirements of different applications, we aim to generalize the reward function. We prefer objective application-specific performance evaluation over subjective application-specific performance evaluation to increase the level of automation without manual expression from the application layer (e.g. Mean Opinion Score). QoS metrics are prominently used in the automated application-specific performance evaluation, thus we utilize WFL (Weber-Fechner Law) [23] and IQX (Exponential Interdependency of QoE/QoS) [23] to calculate reward utility function:

$$\text{qoe}_{\text{i,p}}^{\text{t}} = \gamma_{\text{p}} \times \log(\alpha_{\text{p}} \times \text{qos}_{\text{i}}^{\text{t}} + \beta_{\text{p}}) + \theta_{\text{p}} \quad (2)$$

for positive QoS metrics and

$$\text{qoe}_{\text{i,n}}^{\text{t}} = \gamma_{\text{n}} \times e^{\alpha_{\text{n}} \times \text{qos}_{\text{i}}^{\text{t}} + \beta_{\text{n}}} + \theta_{\text{n}} \quad (3)$$

for negative QoS metrics in which α, β, γ and θ are constant parameters to fine-tune QoS/ application-specific performance relationships. The reward utility function is:

$$req\_utility_i^t = \alpha qoe_{i,p}^t - \beta qoe_{i,n}^t \quad (4)$$

Note that some application-specific performance requirements may conflict with each other (e.g. high Frame-Per-Second (FPS), low playback latency), thus we utilize α, β as the weighting factor controlled by the specified requirement priority from the application.

Moreover, we propose to utilize the soft actor-critic based RL [22] approach which optimizes a stochastic policy in an off-policy manner. The actor-network controls the transport-layer configuration decisions/actions, and the critic network evaluates and provides feedback on the chosen decisions to update the transport-layer policy. Soft actor-critic [22,25] allows our intent-aware transport-layer services to explore the continuous action space of cwnd while being more sample efficient and more robust to brittleness in convergence compared to other approaches.

Regarding other components in the DRL network structure, the input is the state of transport properties and current QoS/QoE, and with two fully connected layers, we also utilize long short-term memory (LSTM) [24] before feed parameters to pre-training actor and critic networks. This allows our intent-aware transport-layer services to make time-series predictions in the environment of a large input space [9]. In actor-critic based approach, the actor-network controls how the end host behaves by learning the optimal policy from a given state as input and aims to make the best transport-layer configuration decisions.

Table 1: Intent-aware transport-layer services' pseudo code

```
When connection is established:
if no entry in intent map
  original transport protocol is used
else if policyCachingBase.contains(entry)
  get transport-layer action from policy caching base
else
  listReq = intentMap.get(applicationId)
  for each requirement in listReq do:
    updateIntentUtility(requirement, requirement.priority)
  end for
  LSTM as function approximator:
    initial policy parameter, Q-function parameters, empty replay buffer
    set target parameters
  repeat
    updateIntentRequest()
    appReqStates = {u}
    transportActions = {a}
    select transport-layer action from a transport-layer policy
    observe next state s', reward r and store (s, a, r, s') in replay buffer
    while update
      randomly sample a batch of transitions (s, a, r, s') from replay buffer
      compute target for Q-functions
      update Q-functions and transport-layer policy by one step of gradient descent
      update target network
    end while
  until convergence
```

The critic network evaluates the action by computing the value function. We utilize soft actor-critic which makes use of three functions: a state value function V, a soft Q-function Q, and a policy function π. We train the three function approximators in line with [22, 25] for discrete transport-layer protocol and CC selections. For continuous transport-layer adaptive configuration, the three function approximators are trained as in [22, 25].

We provide the pseudo-code for our intent-aware transport-layer policy manager in Table 1. First, it updates all the network functions during each epoch in an experience-replay manner. After the actor-critic based training, the actor network can be used to make transport-layer configuration decisions. More specifically, the process consists of two phases: 1) Offline training: the actor and critic networks are built and pre-trained with a number of historic transition samples in order to achieve relatively good initial parameters for phase 2. 2) Online control: start with a set of parameters initialized in phase 1, in each epoch *t,* the agent observes the state s_i^t and obtains the Q-value from the actor-critic networks. Then, a list of action a_i^t are selected based on π-policy, whether to choose a specific transport protocol, congestion control or to increase and decrease congestion control window to a certain amount. The transport-layer policy manager is encouraged to explore different possible actions that assign equal probabilities to actions that have relatively similar Q-values. After the action a_i^t is executed, the agent observes the reward r_i^t and next state s_i^{t+1} on which the action policy keeps updating for the next epoch time *t+1*. The transition (s_i^t , a_i^t , r_i^t , s_i^{t+1}) is stored in the memory at the end of each time period. Note that even though the future transport-layer services should be able to holistically adapt against both user-intended requirements and dynamic network conditions which will result in rapid exploration of huge action space and

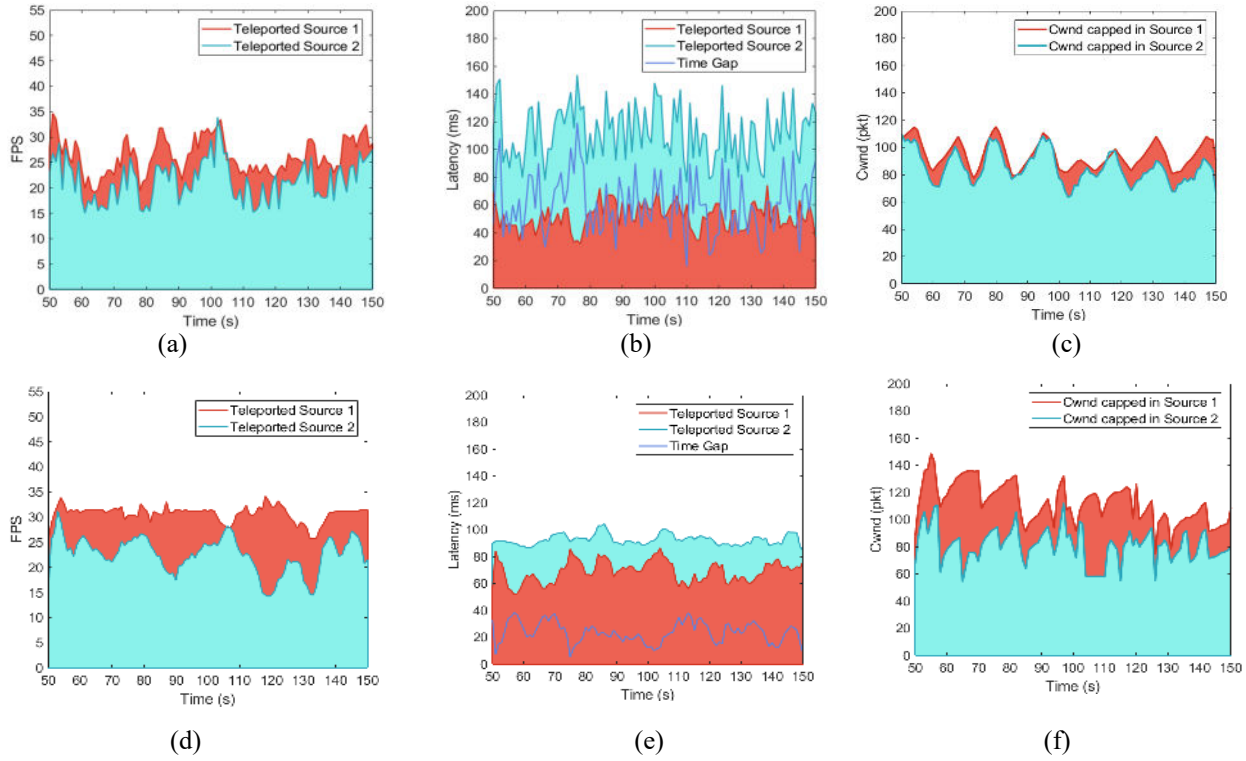

Figure 3. Live volumetric streaming performance with: TCP without intent awareness (a-c) and intent-aware transport-layer services (d-f) in bi-directional multisource live streaming use case which prioritise the frame arrival time.

complexities, in this paper, we focus on the flexible intent of application where there can be multiple users, especially they require for performance Synchronisation, even for the same application type. The intelligent transport layer should be able to understand the actual requirements of the application and adapt its configuration to satisfy them.

IV. Real-life Performance Evaluations

This section presents a multi-criteria evaluation of our intent-aware transport-layer intelligence, focusing on the live volumetric streaming application and its performance with different sets of application requirements and intents.

A. Volumetric streaming application overview

The live volumetric system captures 3D objects from various locations using multiple sensor cameras like Azure Kinect. Each camera sends colour-depth images converted to 3D point clouds to a server. The server receives frames from different clients, processes them, and creates a rendered hologram. This technology enables viewing 3D objects from different angles, making it applicable in various areas such as teleconference, telecommunication, tele-training, entertainment, and healthcare. The paper evaluates the performance of the intent-aware transport-layer services in a predefined scenario: Immersive bidirectional interactions and live teleporting of multiple objects from different network locations prioritize minimizing the time gap between frames arriving at the server from different sources to ensure Synchronisation. The paper mainly focuses on the flexible intent of the application with static but different access delays. It briefly mentions evaluating the transport-layer services in the presence of more complex and dynamic network conditions, leaving the adaptability to future work. Throughout the evaluation, the minimum Round-Trip Time (RTT) of the network is set to 25 ms, and the buffer size is 128 KB.

We design the learning model of the transport-layer policy manager using Python and Tensorflow [26], running on a machine with an Intel i7 3.2 GHz CPU card, GTX 1060 GPU card, and 32GB memory. For parameters of the proposed DRL network. we set the discount factor as 0.99 and the learning rate for both the actor-critic networks is 1e-4. The hidden layers' size is 128. The number of iterations is 10000. We use 70% data for training and 30% of the data for evaluation.

B. Intent-aware congestion window configurations

Then we evaluate the performance impacts of our intent-aware transport-layer services on congestion control configurations (i.e. cwnd) tailoring for the predefined scenario. We evaluate and compare our intent-aware transport-layer services' decision-making against the most popular TCP scheme: TCP Cubic [6], with no intent awareness in its algorithm. In this scenario of live volumetric streaming (see Fig.3), the server receives frames from two different sources: source 1 with 25ms RTT and source 2 with 50ms RTT. Due to the frame Synchronisation requirements, the target is to minimize the time gap between frames coming from different sources with the upper threshold for the time gap being 50ms.
As shown in Fig. 3d-f, our intent-aware transport-layer services could handle not only one-to-one sender-receiver communication but also many-to-one communication for future application usage. In order to reduce the time gap between

frames sent from different sources where source 1 has a shorter path compared to source 2, the proposed transport-layer services are able to adaptively and strictly minimize the queuing delay in the longer path while being more relaxed in the shorter path. At some certain events in source 1, the predicted cwnd is adaptively configured higher than the calculated Cubic cwnd (Fig. 3f) which results in relatively higher playback latency in the shorter path. As a result, this allows the frame time gap between the two sources to keep under 40 ms (Fig. 3e) while both maintain above 20 average FPS performance (Fig. 3d). This comes with the cost of playback latency has been compromised in source 1 (Fig. 3e) in order to achieve the fully synchronized frames from multiple sources [10, 11]. On the other hand, when both sources are running traditional TCP with no intent awareness, the frame arrival time gap varies from 20 ms to 120 ms. The average time gap is 72 ms which exceeds significantly the 50 ms target resulting in severe frame misalignment [11]. In the worst-case scenario, the frame arrival time gap between two clients running traditional TCP Cubic with no intent awareness can go up to 100 ms which is two to three times higher compared to the time gap between the clients running our intent-aware transport services.

V. CONCLUSION

In this paper, we introduce a transport-layer intelligence with application intent awareness, enabling complex applications to express their intent to the transport layer. This innovation supports immersive mixed-reality applications with transparent deployment and configuration of transport-layer protocols. We evaluate our intent-aware services, emphasising frame synchronisation in a multi-party scenario, and demonstrate that our system can autonomously predict the most suitable protocols and configurations to balance complex requirements and achieve application performance targets.

VI. CONTRIBUTION AND ACKNOWLEDGEMENT

Vu San Ha Huynh†, Peng Qian† are co-first authors of this work. This work is funded by SPIRIT project, Grant Agreement 101070672, and the link is https://www.spirit-project.eu/.

REFERENCES

[1] Qian, P., Huynh, V.S.H., Wang, N., Anmulwar, S., Mi, D. and Tafazolli, R.R., 2022. “Remote Production for Live Holographic Teleportation Applications in 5G Networks”. IEEE Transactions on Broadcasting. 2022.

[2] I. Selinis, N. Wang, B. Da, D. Yu and R. Tafazolli. “On the Internet-scale streaming of holographic-type content with assured user quality of experiences”. IFIP networking conference (networking), pp. 136-144. IEEE. 2020.

[3] S. Anmulwar, N. Wang, A. Pack, V. S. H. Huynh, J. Yangy, R. Tafazolli. “Frame Synchronisation for Multi-Source Holograhphic Teleportation Applications - An Edge Computing Based Approach”. IEEE International Symposium on Personal, Indoor and Mobile Radio Communications. 2021.

[4] R. Stewart and C. Metz, “SCTP: New Transport Protocol for TCP/IP,” IEEE Internet Comp., v. 5, n. 6, pp. 64–69. 2001.

[5] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, et al. “The QUIC Transport Protocol: Design and Internet-Scale Deployment”. Proceedings of the Conference of the ACM Special Interest Group on Data Communication. ACM, pp.183–196. 2017.

[6] I. R. Sangtae Ha and L. Xu. “Cubic: A new TCP-friendly high-speed TCP variant”. SIGOPS-OSR, 2008.

[7] N. Cardwell, Y. Cheng, C. S. Gunn, V. Jacobson, and S. Yeganeh. “BBR: Congestion-Based Congestion Control”. In ACM Queue. 2016.

[8] Qian, Peng, Ning Wang, and Rahim Tafazolli. "User Intent Driven Path Switching in Video Delivery-An Edge Computing Based Approach." IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). IEEE, 2022

[9] Volodymyr et al. "Human-level control through deep reinforcement learning. " Nature (2015).

[10] L. Pang, C. Yang, D. Chen, Y. Song, and M. Guizani. “A survey on intent-driven networks”. IEEE Access, pp 1–1, 2020.

[11] T. Pauly, B. Trammell, A. Brunstrom, G. Fairhurst, C. Perkins, P. S. Tiesel, and C. A. Wood. An Architecture for Transport Services. Internet-Draft draft-pauly-taps-arch-12. Internet Engineering Task Force https://datatracker.ietf.org/doc/html/draft-ietf-taps-arch-12 Work in Progress. 2022.

[12] N. Khademi et al., “NEAT: A platform- and protocol-independent Internet transport API,” IEEE Commun. Mag., v. 55, n. 6, pp. 46–54. 2017.

[13] P. S. Tiesel, T. Enghardt, M. Palmer, and A. Feldmann.“Socket intents: Os support for using multiple access networks and its benefits for web browsing”. arXiv preprint arXiv:1804.08484. 2018.

[14] Akshay Narayan et al. “Restructuring endpoint congestion control”. Proceedings of the SIGCOMM 2018, pp 30–43. ACM, 2018.

[15] M. Sardara, L. Muscariello, and A. Compagno, ‘‘A transport layer and socket API for (h)ICN: Design, implementation and performance analysis,’’ Proc. 5th ACM Conf. Inf.-Centric Netw. (ACM ICN), pp. 137–147. 2018.

[16] S. Abbasloo, C. Y. Yen, and H. Jonathan Chao. “Wanna Make Your TCP Scheme Great for Cellular Networks? Let Machines Do It for You!”. IEEE J. Sel. Areas Commun. 39, 1, pp. 265–279. https://doi.org/10.1109/JSAC.2020.3036958. 2021

[17] A. Bahnasse, F. E. Louhab, H. A. Oulahyane, M. Talea, and A. Bakali, ‘‘Novel SDN architecture for smart MPLS traffic engineering-diffserv aware management,’’ Future Gener. Comput. Syst., v. 87, pp. 115–126. 2018.

[18] TM Forum. Tm forum - how to manage digital transformation, agile business operations & connected digital ecosystems.

[19] Network Functions Virtualisation ETSI. Management and orchestration network service templates specification. Technical report, DGS/NFV-IFA014.

[20] ETSI. Multi-access edge computing (mec); framework and reference architecture, 2019. P.

[21] M. A. M. Vieira, M. S. Castanho, R. D. G. Pacífico, E. R. S. Santos, E. P. M. C. Júnior, and L. F. M. Vieira, ‘‘Fast packet processing with eBPF and XDP: Concepts, code, challenges, and applications,’’ ACM Comput. Surv., v. 53, n. 1, pp. 1–36, 2020.

[22] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, ‘‘Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,’’ 2018, arXiv:1801.01290. [Online]. Available: http://arxiv.org/abs/1801.01290.

[23] Reichl, S. Egger, R. Schatz, and A. D’Alconzo, “The logarithmic nature of QoE and the role of the Weber–Fechner law in QoE assessment” . Proc. IEEE ICC, pp. 1–5. 2010.

[24] S. Hochreiter and J. Schmidhuber, ‘‘Long short-term memory,’’ Neural Comput., vol. 9, no. 8, pp. 1735–1780, 1997

[25] P. Christodoulou, ‘‘Soft actor-critic for discrete action settings,’’ 2019, Online].Available: http://arxiv.org/abs/1910.07207.

[26] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, and M. Kudlur, ‘‘Tensorflow: A system for large-scale machine learning,’’ in Proc. OSDI, 2016, pp. 265–283.

[27] Peng Qian, NingWang, and Rahim Tafazolli. Achieving robust mobile web content delivery performance based on multiple coordinated quic connections. IEEE Access, 6:11313–11328, 2018.