QoE Metrics for Interactivity in Video Conferencing Applications: Definition and Evaluation Methodology

Jia He[1], Mostafa Ammar[1], Ellen Zegura[1], Emir Halepovic[2], Theo Karagioules[2]
[1]Georgia Institute of Technology, [2]AT&T Labs – Research

ABSTRACT

Video conferencing applications (VCAs) have become an indispensable tool for business, educational, and personal communications. There is, therefore, considerable interest in understanding and measuring the Quality of Experience (QoE) delivered by VCAs to their users. Video quality, one QoE measure, has received considerable attention in the literature. In this paper, we are concerned with another important aspect of VCA QoE, namely *interactivity*. We define this informally as the ability of a VCA to facilitate satisfying interaction among its users. Interactivity is primarily impacted by the media transmission latency among users which is, in turn, a function of network and application processing delays. Our goal in this work is to address two challenges in investigating interactivity-related QoE in VCAs. First, we propose a suite of meaningful quantifiable interactivity metrics, such as the proportion of silence time and rate of overlapping speech, that correlate well with conversational impairments and, hence, QoE perceptions. Second, we investigate scalable approaches for measuring these metrics. We develop a validated model for user behavior that enables realistic simulation of interactivity in VCA sessions. We also briefly consider an approach to measure interactivity metrics from packet traces. Through a set of experimental results, we demonstrate how our evaluation methodology provides a way for researchers, VCA service providers and network operators to perform large-scale investigations of how latency can interfere with user interactivity and impact VCA QoE.

ACM Reference Format:
Jia He, Mostafa Ammar, Ellen Zegura, Emir Halepovic, and Theodore Karagioules. 2024. QoE Metrics for Interactivity in Video Conferencing Applications: Definition and Evaluation Methodology. In *ACM Multimedia Systems Conference 2024 (MMSys '24), April 15–18, 2024, Bari, Italy*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3625468.3647622

1 INTRODUCTION

Video conferencing has become an indispensable tool for business, educational, and personal communications. Its use has seen a significant increase in recent years. Video conferencing applications (VCAs) are true *multi*media applications, combining several types of media, such as video, audio and various shared content, into an interactive experience. There is, therefore, considerable interest in understanding and measuring the Quality of Experience (QoE) delivered by VCAs to users.

This work was supported by NSF grant NETS-1909040.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MMSys '24, April 15–18, 2024, Bari, Italy
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0412-3/24/04...$15.00
https://doi.org/10.1145/3625468.3647622

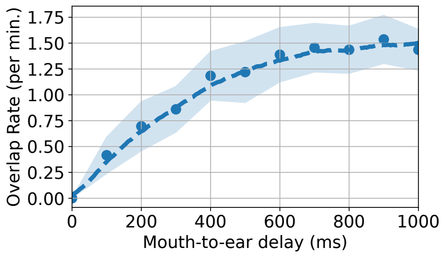


Figure 1: Overlap rate for a simulated conversation between three users with varying mouth-to-ear delay (MED).

QoE is a measure of the satisfaction of users with the services delivered, commonly stated as the "degree of delight or annoyance" [1, 2]. Many aspects of VCAs can influence this satisfaction. One QoE aspect, video quality, has received considerable attention in the literature [3–10]. In this paper we are concerned with another important aspect of VCA QoE, namely *interactivity*. We define this informally as the ability of a VCA to facilitate satisfying interaction among its users. One can argue that interactivity is just as important, if not more important, than video quality in determining user satisfaction with VCAs.

Interactivity is primarily impacted by latency among users; specifically, the Mouth-to-Ear Delay (MED)[1] [11, 12] from every user to every other user in a conference. We presume that ideal interaction occurs when users are in the same physical space where MED is essentially zero (governed by the speed of sound over very short distances). In a VCA running over a typical network context, the MEDs will not be zero. This potentially results in a degradation of interactivity. This work proposes a novel framework to measure such degradation, by introducing a set of interactivity metrics, along with validated techniques to evaluate them.

Changes in MED are exclusively associated with audio media delays within a VCA[2] and, as such, this is the focus of our work. The effect of video mediation on VCA users' interactivity has been shown to vary, ranging from little effect [13–15] to a slight measurable benefit [16–18]. The effect of latency on the ability of video to benefit interactivity in VCAs has been studied [13], concluding that timely delivery of audio is still a critical aspect. Motivated by the highly variable, and possibly negligible, effect that video may have, especially in the presence of latency, as well as the central role audio latency plays in interactivity perceptions, our work focuses on audio delays and their impact on interactivity.

[1]The time from when an utterance is spoken by one user until it is heard by another.
[2]MED can also be increased by the synchronization of audio and video streams. However, this impact will typically remain constant under changing network latency, which will have a uniform effect on both the audio and video.

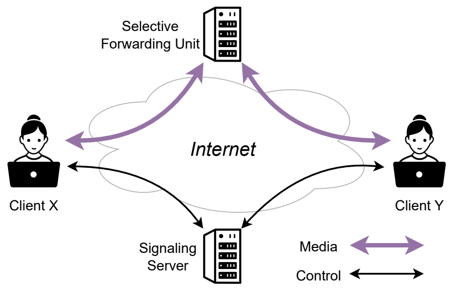


Figure 2: Two-client, server-mediated VCA architecture.

Definitively evaluating interactivity, as with any QoE metric, requires extensive user studies with opinion scoring [11, 19–27]. Indeed, the bulk of existing research assessing interactivity in VCAs has been undertaken by human factors and psychology researchers taking this approach. We survey this literature later in the paper. For now, it suffices to say that such studies are often difficult to perform at scale, and generalize. Our goal in this work is to address two challenges in investigating interactivity-related QoE in VCAs.
1- Defining meaningful quantifiable interactivity metrics. There is a well-understood set of quantifiable metrics (e.g., video bit rate/resolution, stall rate) that proxy video delivery performance and have been shown to correlate well with user-perceived QoE [3–5, 7–10, 28, 29]. This has allowed studies to focus on determining these metrics through end-system monitoring or network packet trace analysis. We aim to develop a similar framework that measures interactivity for VCAs.
2- Developing a scalable and generalizable methodology for evaluating interactivity metrics. We primarily focus on developing a simulation-based approach, based on a model of user behavior that highlights interactivity. We validate the model using results from real VCA sessions. We also briefly explore using packet traces to infer interactivity metrics.

To illustrate what we aim to do, consider one of the metrics that we define: *overlap rate*. This is a measure of how often speakers in a VCA "speak on top of each other". While this can happen in typical in-person conversations, it tends to be exacerbated by increases in the MED between VCA users. Using a simulation method with validated models described later in Sections 3.4 and 4, we simulated a three-client[3] conversation and measured the rate of overlaps as a function of the MED. Figure 1 shows that the overlap rate increases from zero when there is no MED to around 1.4 per minute with one second MED. Our work enables this type of analysis which shows how VCA interactivity is degraded as a function of latency.

The remainder of this paper is structured as follows. In Section 2 we briefly describe VCA structure and components, as well as the sources of latency. We also provide a brief survey of existing literature in the assessment of interactivity from the psychology and human factor communities. We develop the methodology over the next several sections. Section 3 formally develops a set of interactivity metrics and in Section 3.4 we consider alternatives for evaluating these metrics. Section 4 describes our simulation approach and presents the details of the client behavior model. The last step in the methodology is to validate the model in Section 5 using results from real VCA sessions as well as results reported in the literature. Section 6 demonstrates the use of our simulation model in evaluating the interactivity metrics we defined earlier and briefly explores their use for estimating interactivity, based on packet-level information collected via real conversations. It is important to emphasize that the goal of this section is to show the usability of our approach and metrics to provide insights into the interactivity of VCAs and not to provide a comprehensive study of all aspects of VCA interactivity. Section 7 concludes the paper.

[3]We use “client” to refer to a VCA user in the remainder of the paper.

2 BACKGROUND

In this section, we first give a brief overview of typical VCA architectures and enumerate the sources of latency that can impact interactivity metrics within this architecture. We then describe related work in the area of evaluating VCA interactivity.

2.1 VCA Architecture

Commercial VCAs typically allow for operation in one of two modes: peer-to-peer mode or server-mediated [9, 30, 31]. In this work, we focus on the server-mediated mode as it is commonly used even for two-client sessions, and always used in the presence of more than two clients. In this mode, media is forwarded between clients using a server, often referred to as a selective forwarding unit (SFU) when used in this context. In both cases, the client will contact signaling servers [32] to establish a connection to the video conference. The overall server-mediated architecture is shown in Figure 2. Once the connection is established, clients will send media to the SFU. VCAs typically use RTP [33] over UDP for media traffic.

On the client side, the video and audio encoders must compress raw signal data before it can be sent over the network. The encoded video or audio streams must also be packetized. While different VCAs use different encoders, such as H.264 [34] or VP9 [35] for video, and Vorbis [36] or Opus [37, 38] for audio, the overall architecture and hence behavior may be generalized [9]. We note that encoding and decoding only occur at the clients. As typical VCA video codecs such as H.264, VP9, and AV1 support video quality sub-sampling directly on the encoded video data [39–42], the SFU does not need to re-encode the video, which would be a computationally expensive and high-latency task. As a result, the SFU performs minimal processing and is primarily responsible for forwarding copies of received media and appropriately sub-sampling quality in response to network conditions [9, 43, 44].

2.2 Sources of Latency in VCAs

The VCA architecture described in Section 2.1 provides some insights into the different delays experienced by clients. We categorize these into three types: signaling, application, and network delays.
Signaling delays. These are delays that arise while performing a signaling process. This may include joining the conference, unmuting audio, and other similar client interactions with the VCA software. In this paper, we will not consider signaling delays, as they typically do not affect ongoing interactions in a VCA session.
Application delays. The video encoding and decoding process on the clients, including the generation and usage of forward error-correcting (FEC) data, is computationally intensive which leads to additional latency [45–47]. Furthermore, the use of buffers at both endpoints helps to reduce the effect of jitter at the cost of

increased delay [11, 48]. Due to the different operations on sender and receiver, the application delay may also be different on either side. As the SFU performs no decoding or encoding, the processing delays introduced by the SFU are typically negligible when compared to the network delays and client video processing delays. Finally, delays for webcam or microphone capture as well as during video or audio output can be considered as application delays.

Network delays. In the server-mediated mode we consider, this will be the sum of the network latencies from a speaking client to the SFU, and from the SFU to the listening client. The congestion control mechanism implemented by VCAs will typically attempt to keep network latency as low as possible [9, 49].

2.3 Related Work

Prior work on conversation modeling and behavior [21, 50–55] and the use of video-mediated interactions [13–18] have provided insight into how verbal communication between people can be measured and studied in the context of video conferencing. For example, Brady [50] proposes a parameterized model of a two-client conversation over a switched telephone circuit, and how the parameters change as a function of latency. Choudhury and Basu [52, 53] propose a method based on mixed-memory Markov processes [56] to model the influence of people within distributed social interactions. Lai et. al. [21] make use of the AMI Meeting Corpus [57], a large dataset of in-person meeting recordings, to understand the correlation between turn-taking features extracted from the recordings and participant satisfaction with the meeting. The results demonstrate a correlation with a metric known as the turn-taking freedom. In the context of VCAs, Seuren et. al. [55] observe a series of telehealth meetings between a patient and doctor/nurse, providing annotated transcripts depicting the behavior of the clients during problems, such as overlaps or interruptions.

The effect of latency on interactivity on VCAs is primarily based on user studies [11, 19–27]. Each user study involves a task for users, such as constructing a LEGO toy [24], playing charades [25], or ordering pizza [20]. A specific task encourages consistency in the user study and gives context to the results, often collected from a user satisfaction questionnaire [19, 20, 23] or more complex feedback forms [21, 22, 25]. Additionally, Schmitt et. al. [24] construct a QoE model based on features extracted from the LEGO building user study feedback. This model is highly dependent on the specific features that it utilizes. While user studies may provide rich insight from study participants, they are typically limited in scale, and may produce data that is difficult to interpret. We aim to address these potential shortcomings by proposing an alternative method for investigating such effects based on a client behavioral model, alongside a set of well-defined interactivity QoE metrics.

3 METRICS

We now develop a set of VCA interactivity QoE metrics, each individually describing an important aspect of a VCA conversation. The metrics are inspired by the literature survey in Section 2 and describe the quality and usefulness of interaction. One metric, the turn-taking freedom, has been used for in-person meetings in the literature [21], we are now applying it to the context of VCAs.

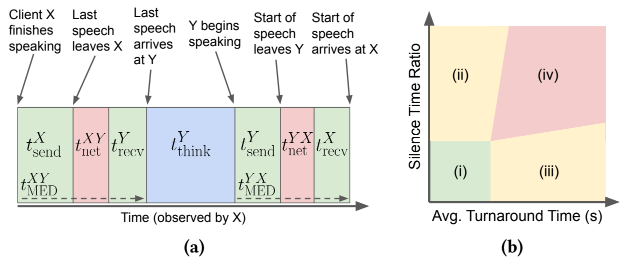


Figure 3: Turnaround time experienced in a VCA: (a) visualization of the various delays making up the overall turnaround time experienced by client X after it finished speaking, (b) relationship between ATT and STR.

We characterize each metric by the conversational aspect it captures and a corresponding mathematical expression. The metrics are categorized according to their applicability to the number of clients in a VCA session: (i) any number of clients; (ii) two clients; and (iii) more than two clients.

3.1 Any Number of Clients

The metrics covered in this subsection are valid for a video conference with any number of clients.

Turnaround Time (TT). This is a foundational measure of client-to-client latency in a VCA. The measure presumes an orderly interaction where one client speaks and another replies. TT in this context is defined as the time from when a client finishes speaking until they begin to hear the response. The components of TT are shown in Figure 3(a). These include application delays at both clients, network delays to and from the SFU and the receiving client's thinking or response time. Formally, we define the turnaround time between client X and client Y, $t_{X\to Y}$, as the sum of several individual delays.

$$t_{X\to Y} = t^X_{\text{send}} + t^X_{\text{recv}} + t^Y_{\text{send}} + t^Y_{\text{recv}} + t^{XY}_{\text{net}} + t^{YX}_{\text{net}} + t^Y_{\text{think}} \quad (1)$$

where t_{send} and t_{recv} are the send or receive application delays at the corresponding client, $t^{(XY)}_{\text{net}}$ is the network delay from client X to client Y, and $t^{(Y)}_{\text{think}}$ is the thinking time of client Y. Because the network path between clients goes through an SFU, the network delay from client X to Y would be equal to

$$t^{XY}_{\text{net}} = t^{XS}_{\text{net}} + t_{\text{proc}} + t^{SY}_{\text{net}} \quad (2)$$

where S refers to the SFU, and t_{proc} is the processing delay of the SFU. In practice, the processing delay is small, due to the implementation details described in Section 2.1.

The quantity $t_{\text{send}} + t_{\text{net}} + t_{\text{recv}}$ is what we referred to earlier as the mouth-to-ear delay (MED) [11], and the expression may be written as below.

$$t_{X\to Y} = t^{XY}_{\text{MED}} + t^Y_{\text{think}} + t^{YX}_{\text{MED}} \quad (3)$$

Note that the MED for an in-person conversation will be zero.

We note that while the individual component delays of TT may be difficult to measure, measurement of the overall TT is relatively straightforward. Practically speaking, this overall delay is the only one that matters, as it is what the clients experience, and typically we will consider the *average TT (ATT)* over some duration.

Silence Time Ratio (STR). The silence time ratio describes the fraction of total conversation time spent without any client speaking. As turn changes are typically silent, the silence time will be lower bounded by sum of the turnaround times normalized by the conversation time. As the turnaround time is dependent on latency, a higher latency implies more time is spent in silence. Additional STR may come from natural breaks in conversation or other events such as "awkward silence". STR is defined as the following:

$$STR \geq \frac{\sum_i t_{X\to Y}^{(i)}}{T}, \tag{4}$$

where $t_{X\to Y}^{(i)}$ is the i^{th} turnaround time, and T is the total conversation time. As with ATT, STR on its own may not give the full picture as to the character of turn-changing in the conversation. However, the combination of ATT and STR as shown in Figure 3(b) provides a stronger context. Each case indicates the following situations: (i) "low" ATT and STR indicates a baseline conversation; (ii) "high" STR and "low" ATT likely indicating a conversation with short speech bursts and a higher rate of turn changing; (iii) "low" STR and "high" ATT likely indicating a conversation with long speech bursts and less frequent turn changes; (iv) "high" STR and "high" ATT indicating a high potential for impaired QoE due to high latency.

Overlap Time Ratio (OTR). The ability of VCA clients to partake in turn-taking depends on their ability to detect the presence of other clients' speech. Latency between clients will hinder the ability to do this in a timely manner, increasing the risk of speaking out of turn, therefore resulting in multiple clients speaking at once. Overlap time ratio (OTR) describes the percentage of total conversation with more than one client talking. In a typical conversation, the scenario of multiple people talking simultaneously causes inefficient communication, as no useful information can be exchanged, and thus OTR can be understood as a measure of the wasted time in a conversation. The definition of OTR is similar to STR:

$$OTR = \frac{\sum_i \tau_i}{T}, \tag{5}$$

where τ_i is the duration of the i^{th} overlap. This is the time from the start of overlapping speech until one client restarts talking. We note that the presence of very short utterances (VSUs, also known as backchannel communication [51]) may also be counted as τ_i. VSUs differ from "overlaps" as they typically do not cause a conversation repair, even though they occur while someone else is speaking. For example, a common form of VSU is a short statement of agreement with what is currently being said.

Overlap Rate (OR). In a typical conversation, repairing overlapping speech may require an action from the participants to reset the current turn. Prior work [54, 55] suggests that overlaps are generally short in duration (a few syllables), and regardless of duration, they result in the same type of repair. As a result, measuring the overlap rate (OR) gives insight into how often this scenario arises during the conversation. OR is defined as:

$$OR = \frac{N_\tau(T)}{T}, \tag{6}$$

where $N_\tau(T)$ is the number of overlaps up to current time T.

Useful Conversation Time Ratio (UCTR). This is the fraction of time that is available for useful information exchange between participants during the video conference. Therefore, the UCTR is defined at the high level as the following:

$$UCTR = \frac{T - \sum_i \tau_i - \sum_i t_{x\to y}^{(i)}}{T} = 1 - OTR - STR \tag{7}$$

which is simply the total conversation time T minus all overlap durations τ_i and turnaround times $t_{x\to y}^{(i)}$ normalized by the total session time. There may also be other delays that may be included in UCTR, such as breaks that occur when a new client joins the video conference.

In general, UCTR combines several effects into one metric. It includes both the effect of longer delays between turns and a higher overlap rate due to higher latency. As a result, UCTR and interactivity QoE are closely related.

3.2 Suited for Two Clients

The metric described in this section is most suited for a video conference with two clients.

Repeat Rate (RR). As video conferences between two people are relatively common and often of high importance, for example, telehealth appointments [55], it may also help to evaluate the repeat rate (RR), which is the likelihood of one of the participants to attempt to repair the conversation by repeating their last statement. As described in [55], if a conversation participant does not hear a response within an expected amount of time, the typical behavior is to either prompt for a response again or repeat what was said last. Either of these leads to wasted communication time, as no new information is exchanged. Adding latency between the two clients will impact the ability to detect when the other client begins their turn, so a higher latency will lead to a higher RR.

This metric is of particular importance in the two-client case given the expectation of frequent turn-taking. RR is defined as:

$$RR = \frac{C_R(T)}{T} \tag{8}$$

where $C_R(T)$ is the count of repeats up to time T.

3.3 Suited for More than Two Clients

This metric is only suited for a video conference with more than two clients.

Turn-taking Freedom (TTF). This describes the ability of each client to take a turn. It considers both the number of turns taken by each client, as well as the order in which the turns are taken. If the allocation of turns is balanced and the order is mostly random, then the TTF will be higher. The value of TTF was found to correspond to user satisfaction when calculated using in-person meeting recordings [21], and so a higher TTF should correspond to better QoE.

TTF is defined in the literature [21] as the following:

$$TTF = 1 - \frac{H_{max}(Y|X) - H(Y|X)}{H_{max}(Y|X)}, \tag{9}$$

where H is the conditional entropy of turn-taking, and H_{max} is the theoretical maximum of this value, determined by the number of clients. The expression for conditional entropy is:

$$H(Y|X) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\cdot \log p(y|x), \tag{10}$$

where $p(x, y)$ is the joint probability of client Y taking a turn after client X and $p(y|x)$ is the corresponding conditional probability given X finished their turn. These probabilities can be sampled by empirical measurement.

A high $H(Y|X)$ leads to a high TTF. As entropy describes the uncertainty of a random variable, high $H(Y|X)$ intuitively means that guessing the turn order is difficult, which can only be the case if all clients have a fair chance to take a turn.

3.4 Evaluation of Metrics

There are three primary methods available for evaluating the interactivity metrics: (i) meeting recordings, (ii) packet traces, and (iii) simulation. Each metric can be measured by any of the three methods. However, there are important differences in the practicality and scalability of each. We utilize each of these methods in the subsequent sections of the paper.

Meeting Recordings. Meeting recordings have been used in prior work [21, 53] to study the effect of certain conversation characteristics on user satisfaction. They contain detailed information about a conference, enabling the detection of more complex conversational scenarios such as interruptions. On the other hand, meeting recordings are difficult to scale. For research purposes, meeting recordings are typically collected within the context of user studies, with privacy concerns making the widespread collection of recordings impossible. Recordings are also typically not available to organizations such as network operators. Finally, extraction of interactivity ground truth and reconstruction of conversations from recordings can be a challenging task [53, 58].

Packet Traces. Packet traces are a more scalable alternative to meeting recordings, but they collect network data traffic rather than actual conversations. They are used to investigate the characteristics of network flows and typically available to network operators or VCA providers at various points in their networks. Analysis of packet traces has been used in the study of video quality metrics for video streaming QoE [29] and video conferencing [10]. Recent work has shown how to decode unencrypted header structures for some VCAs [59] offering considerable insight into the conversation from the packet trace alone. As described in more detail in Section 6.3, there are several measurable parameters which vary as a function of the client's microphone input, including the audio packet rate and size, and corresponding audio bitrate. These time-varying values may be used to detect conversational patterns.

Simulation While meeting recordings and packet traces can provide information relating to client interactivity, they still rely on the collection of data, making scalability a concern. Instead, we consider simulating VCA sessions as a means to develop insights into interactivity metrics and how they are impacted by latency. A simulation-based approach demands a behavioral model of a VCA client that captures as much behavior as possible whilst being general enough to be used in many different simulation scenarios. To this end, we present a client model in Section 4, which we employed in various simulation scenarios in Section 6.

4 SIMULATION-BASED METRIC EVALUATION

In this section, we develop a simulation-based method that can be used to evaluate the interactivity metrics from Section 3. We describe the simulation setup, followed by the client behavioral model, the core part of the simulation method. The simulation method and user behavioral model are validated in Section 5, using data collected from real conversations conducted over a popular VCA as well as values from literature.

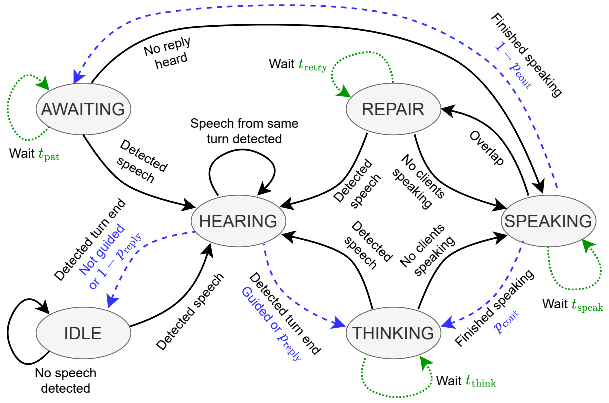


Figure 4: Client state machine used for simulation.

4.1 Simulation Setup

The simulation setup includes two main components: the client behavioral model, described in Section 4.2, and the network simulation. In our work we focus on the user behavioral model and the effect of its parameters on interactivity metrics experienced by the user. For the network simulation part, any of a number of approaches can be used, e.g., NS-3 or Mininet. However, in order to focus on validating our user behavioral model, we do not use a full network emulator at the packet-level in our results. Rather, we use a simple network simulation allowing for constant latency to be defined between all pairs of clients. This latency may be chosen so as to include latency on the SFU path as described in Section 2.

The simulation outputs a list of states occupied by each client at each simulation step, every 10 ms in our implementation. The simulated conversation time can be any length, but we consider simulated conversations between ten and thirty minutes. We implemented the simulator in Python, making use of the multiprocess library, allowing for hundreds of "hours" of simulation to be completed in several minutes of real time. This performance was achieved on a 2018 Macbook Pro laptop. The fact that no high-performance infrastructure is needed for simulation is an important aspect as our model design prioritizes scalability over complexity.

4.2 Client Behavioral Model

Our simulation approach relies on a client model which approximates the behavior of a real video conferencing participant. In designing the model, we aim to balance behavioral coverage while simultaneously keeping the model simple, as this will aid the goal of scalable experimentation. The foundational behavior that we attempt to capture with the model is turn-taking as described in literature [54, 55]. We further take inspiration from the carrier-sense multiple access with collision detection (CSMA/CD) protocol [60, 61], as the video conference represents a shared communication medium and some metrics, such as UCTR, can be understood as analogous to goodput in a CSMA/CD network [62–65].

Table 1: List of parameters for the video conferencing client model. Parameters marked with an asterisk may be changed to define the specific client behavior.

Parameter	Purpose	Value
t_{speak}	The duration of a client's speech during their current turn.	$\sim$ Unif($t_{\text{smin}}, t_{\text{smax}}$)
t_{pat}	The time taken by a client to wait for a reply (the client's patience).	$\sim$ GIGa(P, 0.4)
t_{retry}	The backoff time for a client after detecting an overlap	$\sim$ Exp(λ)
$t_{\text{think}}{}^*$	Represents the thinking time of a client before starting to speak.	$\sim$ GIGa(1, 0.4)
$t_{\text{smin}}{}^*$	Minimum client speaking time	Dependent on conversation scenario, typ. 3-30 s
$t_{\text{smax}}{}^*$	Maximum client speaking time	Dependent on conversation scenario, typ. 10-90 s
P^*	The average value for t_{pat}.	3 s initially, value is learned over time
λ^*	Inverse mean of t_{retry}	Dependent on conversation scenario, typ. 0.2-1
$p_{\text{reply}}{}^*$	Probability that a client tries to reply after a client finishes speaking.	1 if two-client conversation, 0.1-0.8 otherwise
$p_{\text{cont}}{}^*$	Probability that a client continues its turn after another thinking time.	Typically < 0.1
$p_{\text{guide}}{}^*$	Probability that a client selects another client for the next turn.	$0 - 1$

We choose to implement the model as a state machine, which simplifies the representation by ensuring that the client can only exist within a finite set of states. Transitions between states are event-driven, helping to define a set of behaviors that approximate real clients. The state machine representation of the client is shown in Figure 4. The precise behavior of the model is further specified through the set of parameters in Table 1, which allows for the generation of different conferencing scenarios as in Section 6.

Altogether, the client model captures a set of behaviors, including (i) turn-taking with a mix of random and deterministic characteristics, (ii) overlaps caused by multiple clients attempting to reply to the same turn, and (iii) waiting to hear a reply before attempting to continue a turn. In the following subsections, we first describe the client parameters, followed by the client states. We then briefly describe the simulation implementation.

4.3 Client Parameters

The client parameters are enumerated in Table 1. Altogether, these parameters tune the overall client model behavior and may be used to generate certain "types" of client and therefore conversation scenarios. There are three different types of parameters in the table.

The first type is time delays, which define when certain state transitions occur. Delays are chosen according to various distributions. For example, t_{pat} and t_{think} are distributed according to the generalized inverse gamma (GIGa) distribution, shown to closely match empirical measurement of human reaction time [66, 67]. t_{retry} is exponentially distributed, matching state transition timeouts in literature [50]. The second type of parameter defines the shape of these time delay distributions. These include P and λ which define the average of t_{pat} and t_{retry} respectively, and t_{smin} and t_{smax} which define the minimum and maximum values for t_{speak}.

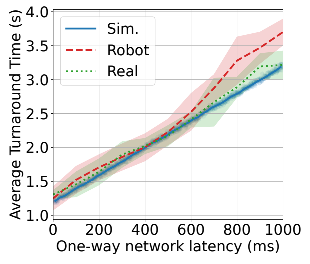
The last three parameters in the table are of the third type. These parameters define the probability of certain binary-choice transitions in the client model. These three parameters are highly consequential for defining the behavior of a client. Examples of clients with different parameter values are provided in Section 6.

4.4 Client States

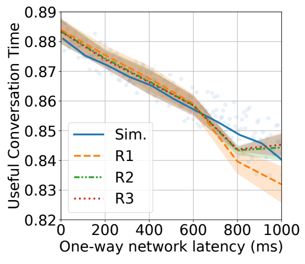
The client model in Figure 4 contains six states.
Idle. This is the starting state for all clients other than the initial speaker. In Idle, the client will be listening for when another client begins speaking. Once speech from another client is detected, the client will transition to the Hearing state.
Hearing. The client can hear speech from another client in this state. The client will remain in Hearing until it detects that the client has stopped speaking. Once this event is detected, there are two possible outcomes; (i) the client returns to Idle or (ii) the client begins its turn and transitions to Thinking. Case (ii) only happens if the client was guided to speak by a prior speaking client, or if the floor was left open and the p_{reply} probability check was successful.
Thinking. When the client enters this state, the t_{think} timer begins counting down. The Thinking state represents the natural thinking time that a client has between hearing the end of a turn and beginning to speak. Once t_{think} elapses, the client will check if anyone else is currently speaking, and if not, go to the Speaking state. If another client is speaking, then the client will re-enter the Hearing state. This is equivalent to the carrier sense scheme in CSMA/CD.
Speaking. This state represents a client in the process of speaking, and is the starting state for the initial speaker. The client remains in Speaking until the t_{speak} timer elapses, at which point there is a probabilistic transition determined by p_{cont}, according to which the client returns to Thinking and attempts a turn continuation or transitions to Awaiting. While in Speaking, The client will continuously monitor if another client also enters the Speaking state. If this occurs, the client will exit the Speaking state immediately and transition to the Repair state. This represents an overlap scenario.
Awaiting. This state is used to await a response. It enables a client to continue its turn if it does not hear a reply within the t_{pat} timer. If the timer expires before hearing a reply, the client will return to the Speaking state. If the client detects a reply while the t_{pat} timer is running, it will return immediately to the Hearing state.
Repair. All clients speaking in parallel will enter this state once they detect an overlap, similar the recovery after collision detection in CSMA/CD. Clients remain in Repair until all clients have stopped speaking and the t_{retry} timer has elapsed, similar to CSMA/CD's backoff time. At this point, the client returns to Speaking if it detects no other speaking clients. Otherwise, the client returns to Hearing.

4.5 Discussion

Adaptive Behavior. As delays in video conferencing have been found to influence a client's perception of others [23], VCA clients can mitigate issues by adjusting their behavior [26, 55]. In the client



(a)



(b)

Figure 5: Experimental model validation results. (a) comparison of measured ATT for 2-client case, (b) comparison of measured UCTR for 3-client case.

model, this behavioral adjustment is implemented in two ways. The first is through the use of an adaptive patience value P. The adaptation is done based on a running average of the most recent six turnaround times as measured by the client during simulation. The second way is by increasing t_{retry} exponentially after consecutive overlaps, similar to the backoff time in CSMA/CD.

Potential Model Extensions. The client model presented above implements several key client behaviors, however, it is not an exhaustive set of all behaviors experienced in real VCA settings. In particular, the client model does not capture purposeful interruptions or interjections. The decision not to include purposeful interruption is motivated in two ways: first, we wish to avoid over-complicating the model, as this harms the goal of scalability, and second, purposeful interruption is typically regarded as impolite or unusual behavior, making it difficult to validate from existing literature if it were also included in the model. We remark that a simple model of purposeful interruption could be an exponentially distributed timer started by each client whenever a new turn begins.

5 MODEL VALIDATION

In this section, we validate the client model from Section 4 in two ways: (1) we compare model simulation output to meeting recordings captured by a VCA, and (2) we relate real conversations studied in literature to simulation output captured under similar conditions.

5.1 Validation from VCA Meeting Recordings

We validate a subset of the client behavior using meeting recordings from a popular VCA. In particular, we validate the client model operation without the possibility of overlaps or timeouts in the Awaiting state. This allows us to develop a set of baseline experiments with a VCA involving both real users and robot users. The robot users access the VCA the same way as real users, but their behavior is defined according to a subset of the client model described in Section 4.2. The robot users send and receive real audio through the VCA and take turns automatically. One-way network latency between 0 and 1000 ms is added between clients in both the meeting recordings and simulation. We measure application latency from the VCA (caused by video encoding/decoding and buffering at the users) at around 100 ms. This is added to the network latency, creating the total latency between clients.

5.1.1 Experimental Setup. With two-clients, we use a scripted conversation for both the real and robot users, lasting close to

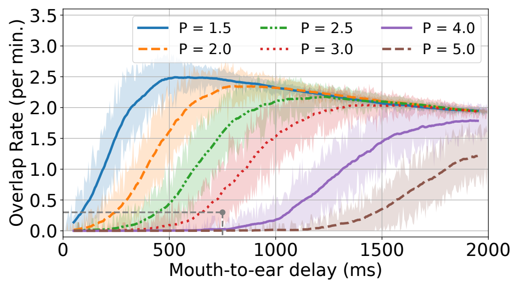


Figure 6: Two-client simulation of OR as a function of MED used for model validation. Operating point from literature [55] is marked by a gray point.

two minutes. For the simulation, we set the client parameters to closely resemble the scripted conversation. Specifically, t_{smin} and t_{smax} are set such that the average value is the same as the average length of the scripted conversation turns. To replicate the scripted nature of conversation in simulation, we set $p_{\text{reply}} = 0$, $p_{\text{cont}} = 0$, $p_{\text{guide}} = 1$, and $P = \infty$. For this experiment, we compare the Average Turnaround Time (ATT) calculated with meeting recordings from a popular VCA to the ATT calculated from the simulation output.

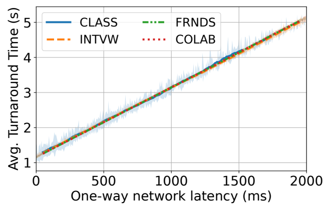
A similar experimental setup is used for the three-client case, except that only the robot speakers are used, and latency is only added to the uplink and downlink of a single client. The simulation client model parameters are set in the same way as they are in the two-client case. We evaluate the Useful Conversation Time Ratio (UCTR) in the three-client case to provide validation of a different metric. In all cases, the real/robot experiments are repeated ten times for each latency evaluation, as are the simulations.

5.1.2 Results. The results of the two- and three-client validation experiments are shown in Figure 5. Each dot for the simulation results represents the average over the repeated runs for the given latency, and the line is a windowed average over 20 points. The real VCA results are presented as an average with shaded standard deviations. The two-client simulation in Figure 5(a) is generally within the standard deviation of the robot and real experiments, giving confidence in model accuracy. There is a slight deviation in the robot experiments for ATT at higher latencies. We determined this to be caused by experimental artifacts, namely the nondeterministic processing delays introduced by the audio processing libraries used for the robot speakers.

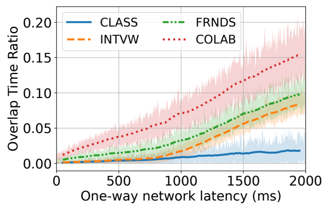
Similarly, the spread of three-client simulation results in Figure 5(b) is within the standard deviation of the robot clients R1–R3, though we note some difference at higher latencies for this experiment as well. We conclude from these results that the subset of the simulation model producing the general turn-taking behavior is reasonably accurate compared to conversational turn-taking on real VCAs, for cases with two clients and more than two clients.

5.2 Validation from Literature

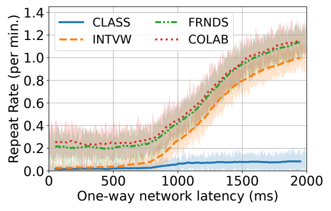
Other behaviors supported by the client model can be validated through existing literature involving user studies. We are particularly interested in the ability of the simulation model to correctly capture overlap behavior, as this is the primary “problem” that the simulation model attempts to capture.



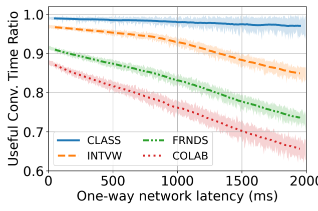
(a) ATT



(b) OTR



(c) RR



(d) UCTR

Figure 7: Simulation results for the two-client conversation scenarios.

Table 2: Client parameters used in the four two-client simulation scenarios. Each parameter is specified for each client.

$\{C_1, C_2\}$	INTVW	FRNDS	COLAB	CLASS
t_{think}	$\{\sim \text{GIGa}(1, 0.25), \sim \text{GIGa}(1, 0.25)\}$			
t_{smin}	$\{5, 30\}$	$\{6, 6\}$	$\{4, 4\}$	$\{120, 5\}$
t_{smax}	$\{15, 90\}$	$\{18, 18\}$	$\{12, 12\}$	$\{240, 15\}$
P	$\{3, 3\}$			$\{5, 3\}$
λ	$\{0.5, 1\}$	$\{1, 1\}$		$\{0.5, 1\}$
p_{reply}	$\{1.0, 1.0\}$			$\{1.0, 0.5\}$
p_{cont}	$\{0.1, 0.1\}$		$\{0.25, 0.25\}$	$\{0.9, 0.1\}$
p_{guide}	$\{0.0, 0.0\}$			

5.2.1 Simulation Setup. To evaluate the correctness of overlap behavior, we use a two-client simulation with the possibility of overlap, which is enforced by setting $p_{\text{cont}} = 0.1$, and finite values for the patience, P, which do not adapt as in Section 4.5. The distribution of t_{think} is set to $\text{GIGa}(1, 0.25)$ such that the average is one second. As P is a critical value for the likelihood of overlap in the two-client case, we test different values and observe the outcomes. The two clients have an average speaking time of 20 seconds.

Latency values between 0 and 2000 ms are added between the two clients, and an application latency of 100 ms is also included. This is typical of the latencies considered in the literature and encountered in regular VCA use on typical home, cellular or enterprise networks. We take 25 ten-minute simulations per tested latency value, chosen to represent a shorter two-person conversation, and calculate the overlap rate (OR) as the metric for this experiment.

5.2.2 Results. The results of this simulation are shown in Figure 6, for patience values P between 1.5 and 5 seconds, showing the average and standard deviations. We observe that Seuren et. al [55] measure an average of 0.3 overlaps per minute over the duration of their study, while mentioning a 750 ms end-to-end latency (MED) for the clients. The results in Figure 6 show that at MED = 750 ms, an OR of 0.3 is within the standard deviation of the simulation results with $P = 3$. Furthermore, this is close to the 3–5 overlaps per minute recorded by Egger et. al [20] given the considerably shorter turn lengths under similar latency conditions. While a slightly higher patience value would align better to the literature result, we set $P = 3$ as the default parameter value to avoid an over-specification of the model. We note that the decrease in overlap rate (OR) at higher MED is likely due to the increase in turnaround time, meaning fewer turns are taken in a given duration.

6 RESULTS AND INSIGHTS

In this section, we will present a series of simulation experiments using the client model. We do not aim to provide a comprehensive study of the interactivity metrics and how they are impacted by network and system latencies. Rather, the goal is to demonstrate the capabilities of the scalable simulation approach in providing a framework for investigating interactivity in VCAs. Extensive studies using our metrics and methodology are relegated to future research. First, we investigate a set of two-client conversations, using scenarios that are typical of how VCAs would be used. Second, we investigate a larger conversation scenario with ten users. To the best of our knowledge, VCA conversations at this scale have not been studied in detail in the literature. Finally, as active measurement may be able to provide enhancements to our approach, we present a brief example and discussion of such methods applied to real conversations held on a VCA at the end of this section.

6.1 Two-Client Scenarios

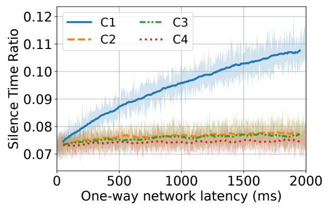
We chose a set of four conversational scenarios representing common uses of VCAs in practice. For each scenario, the two clients are given model parameters implementing the scenario behavior, summarized in Table 2 and explained further below:

Interview (INTVW). This scenario represents a typical interview scenario occurring over a VCA. One client has the role of the interviewer, who will take shorter turns on average (10 s), and the other client is the person being interviewed, who will take much longer turns (60 s on average). Furthermore, the interviewer has a longer backoff time by setting $\lambda = 0.5$, representing the tendency for the interviewer to let the other person speak in case of an overlap.

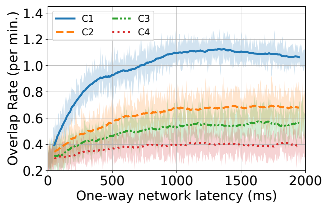
Friends (FRNDS). This is a conversation between friends. Both clients are defined identically, each with short speaking times, making this scenario a relatively fast-paced conversation.

Collaboration (COLAB). In this scenario, the clients are involved in a remote collaborative exercise, for example fixing a machine or a piece of software code. The clients are defined identically, with short speaking times. p_{cont} is set to 0.25, larger than other scenarios, representing the increased likelihood of turn continuations.

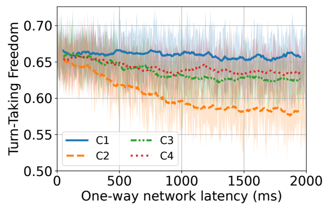
Class. The parameters used by the client model allow it to represent groups of individuals, for example, a class audience as a whole listening to the other client (the instructor) teaching. While a simulation with individual client models representing each member of the audience would be possible, modeling as a single client can help reduce complexity and simulation time. For this scenario, the instructor client is given very long speaking times and a very high probability of continuing a turn, reflecting the conversational patterns while teaching. The class "client" has only a 50% chance of replying at the end of a turn, representing the sporadic nature of questions that are asked by the class. Furthermore, the instructor client has a high value for P and a low value for λ, representing the nature of an instructor to give way to a student asking a question.



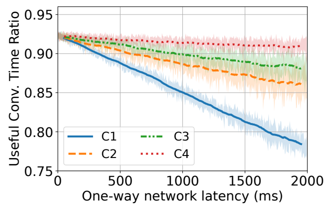
(a) STR



(b) OR



(c) TTF



(d) UCTR

Figure 8: Simulation results for the ten-client conversation scenario.

6.1.1 Simulation Setup. For each of the four conversation scenarios, we use the implementation as described in Section 4.1 to simulate the client models defined in Table 2. One-way latency between the two clients is varied between 0 and 2000 ms. An application latency of 100 ms is also added between each client, representing the sum of t_{send} and t_{recv} from Equation 1. For each measured latency, we run 25 simulations, each for 10 minutes of simulated time. The simulation output is used to calculate the six metrics from Section 3.

6.1.2 Results. The results for the interactivity metrics as a function of the applied latency are provided in Figure 7. Due to space limitations, we report the results for ATT, OTR, RR, and UCTR only, though the other metrics may be calculated from the same results.
ATT. Figure 7(a) shows that the ATT for all scenarios is the same as a function of the additional latency, increasing from 1.25 s to 5.25 s. This is the expected result, as the parameters in Equation 1 are the same for all four scenarios, including the thinking time.
OTR. OTR is generally low for all scenarios with zero additional latency. OTR increases quickly for COLAB compared to the other scenarios, likely due to the increased value for p_{cont} in this scenario. OTR for the INTVW and CLASS scenarios is similar up until the 1,000 ms point, where the OTR for INTVW begins to increase at a much higher rate, likely due to the more frequent turn changes. The trend for FRNDS is similar to INTVW, but with slightly higher OTR across all additional latencies.
RR. RR stays constant up to around 1000 ms additional latency, where the initial patience value $P = 3$ becomes insufficient. At this point, the client's t_{pat} is more likely to time out before hearing a reply, causing the client to continue speaking again. There are between 1 and 1.2 repeats per minute for the FRNDS, INTVW and COLAB scenarios at the highest latencies.
UCTR. The four conversations demonstrate significantly different trends for the UCTR. CLASS is practically unaffected by latency, while INTVW is only lightly affected up until around 1000 ms. The FRNDS and COLAB scenarios are severely affected by the latency, dropping to below 70% and 60% UCTR at the highest applied latencies, respectively. These results show a heavy dependence on the conversation type when considering the effect of latency.

These results for the two-client scenarios show a heavy dependence on the particular use case of the VCA. From the point of view of the network operator, the use case may be considered actionable information, possibly affecting how latency is managed through load balancing or customer subscription plans. Operators may also find valuable certain latency values as thresholds for good vs. deteriorating customer experience. In addition, the VCA itself may benefit from knowledge of the use case, which may help optimize the use of multiple interfaces [68].

Table 3: Parameters for the ten-client simulation scenario.

	C_1	C_2	C_3	C_4–C_{10}
t_{think}	GIGa(1, 0.25)	GIGa(1.25, 0.25)	GIGa(1.5, 0.3)	
t_{smin}	10			
t_{smax}	30			
P	3			
λ	4	0.25		0.1
p_{reply}	0.8	0.5		0.0
p_{cont}	0.2		0.1	
p_{guide}	0.8		0.2	0.0

6.2 Ten-Client Scenario

For the ten-client case, we use a scenario representing a typical meeting held on a VCA. One client is leading the meeting, two clients are responding actively to the leading client, and the other clients are idle, only speaking if prompted by one of the three active clients. In the authors' experience using VCAs, this is a very common scenario for using a VCA with a larger number of clients.

Table 3 lists the client parameters used for the ten-client scenario. These parameters produce the leader C_1, with the highest p_{reply}, p_{cont} and p_{guide}. C_2 is the second most active, followed by C_3. The main difference between C_2 and C_3 is the lower values of p_{cont} and p_{guide} for C_3, meaning that C_3 is less likely to continue its turn and less likely to guide other clients to speak. Lastly, setting $p_{\text{reply}} = 0.0$ for the remaining clients ensures they remain inactive unless guided by C_1, C_2, or C_3.

6.2.1 Simulation Setup. We use the ten-client scenario in four ways by adding latency to one of C_1, C_2, C_3, or C_4. This represents a common scenario for excessive latency where it affects only one client. As in the two-client scenarios, we simulate additional one-way latency between 0 and 2000 ms and include the application delay. For each tested latency, we run ten simulations, each for 30 minutes of simulated time. We calculate the six valid metrics from Section 3, this time substituting TTF for RR.

6.2.2 Results. Figure 8 presents the results for STR, OR, TTF, and UCTR for the ten-client scenarios.
STR. The STR increases from around 7.5% to 11% when the latency is added on client 1, meaning the total silence time almost doubles. However, when the latency is added to the other clients, there is only a small decrease in the STR, likely due to the dominance that client 1 has during this conversation scenario.
OR. The OR for all four latency scenarios follows the same trend, an initial increase followed by a leveling off. This indicates that overlaps occur every time the client attempts to speak, and this situation happens when the latency applied to a client exceeds the average thinking time of the client. Under these latencies, the client

is not able to detect another client speaking before it chooses to reply, causing an overlap. When very high latency is added to client 1, OR decreases, likely due to fewer turns taken in a given period.

TTF. The TTF is shown to either stay constant or decrease, depending on which client is affected by high latency. If the latency is applied to client 1, the TTF remains constant, most likely because the additional overlaps will still be resolved in client 1's favor due to its high value of λ. If client 2 is affected by the latency, the TTF drops significantly, likely because client 1 has an even higher chance of taking a turn during an overlap. If client 3 or client 4 experiences latency, the TTF drops slightly.

UCTR. Adding the latency to different clients has a significant impact on the UCTR. Depending on which client it applies to, high latency can cause the UCTR to vary by up to 17 percentage points. This also shows that if one of the inactive clients experiences excessive latency, there is only a minor effect on the UCTR.

6.3 Evaluation with Packet Traces

While simulation provides a scalable method to measure different conversation scenarios with various numbers of clients, methods to analyze real data are still important. Real-time measurement of user QoE has been of interest to network operators for applications such as video streaming [29, 69, 70], with techniques extending to video conferencing [10, 59, 71]. In this subsection, we outline a method for estimating UCTR with VCA packet traces.

6.3.1 Method. The evaluation in this section uses packet traces collected on the client side. As described in Section 3, there is an effect on the TT metric depending on where the packet trace is collected. In this section, we focus on the UCTR, which will not be affected by this detail, and thus can be measured in the same way from packet traces collected within the network.

The analysis method relies on the extraction of values from the packet header using the method outlined in [59]. Specifically, we filter out audio packets using the header value for "packet type". Secondly, we filter out packets sent or received by each client using the RTP synchronization source (SSRC) also found in the multimedia stream packets. The SSRC uniquely identifies clients within a meeting [33, 59]. Using controlled experimentation under the standard audio settings used by the VCA, we observe that the audio packet sending rate correlates with the client's microphone input. The VCA will send audio packets at a rate of 50 per second with active microphone input, and a rate of 10 per second otherwise.

A more significant difference between audio detected or not detected is observed when considering the ratio of upload to download rate. Therefore, using the filtered audio packets, we calculate this ratio for a given client X with the following expression:

$$R_X(T) = \frac{\sum_{t\in[T,T+\Delta t]} U_X(t)}{\sum_{t\in[T,T+\Delta t]} D_X(t)} \tag{11}$$

where $\Delta t = 0.25$ s, $U_X(t)$ is the size of an upload packet sent by client X at time t, and $D_X(t)$ is the size of a download packet received by client X at time t. This produces a function $R_X(T)$ sampled at 0.25 s intervals. The value of Δt is informed by the limited audio packet rate, which is as low as ten per second. A smaller value for Δt leads to a noisy estimate of $R_X(T)$, whereas a larger value of Δt leads to imprecise measurement of the interactivity metrics. Having

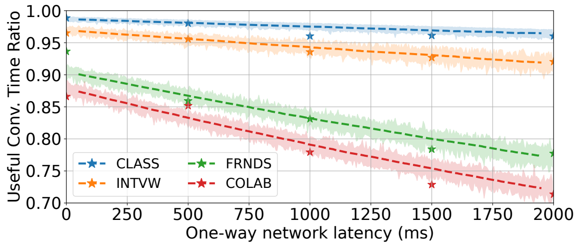


Figure 9: UCTR comparison between simulated scenarios from Section 6.1 without overlaps and measurements of VCA packet traces for equivalent real conversations (dots).

calculated $R_X(T)$, the presence of client X's speech is determined by comparing $R_X(T)$ to a threshold; we find that a threshold of $R_X(T) > 2$ indicates that client X is speaking, $R_X(T) < 0.5$ indicates that client X is hearing speech from another client, and $R_X(T)$ otherwise indicates either silence time or overlap time. Given the analysis method, estimation of the UCTR is possible by simply measuring the total time with $R_X(T) > 2$ or $R_X(T) < 0.5$.

6.3.2 Results. We evaluated the method by comparing simulation results to packet traces recorded from controlled experiments. In particular, we simulated the four scenarios from Section 6.1 without the possibility of overlap, and recorded packet traces from VCA sessions where clients conducted a conversation closely resembling the simulated conversation. Figure 9 shows that the UCTR measured from the VCA packet traces (represented as stars in the figure) generally correspond closely to the UCTR measured from simulation. With the exception of some points (FRNDS conversation, zero additional latency and COLAB, 1,500 ms additional latency), all packet trace measurements of UCTR fall close to the standard deviation of simulation results for each conversation scenario.

Given these results, the packet trace method as outlined in Section 3.4 has been shown to be a viable method of estimating interactivity. Alternative methods of identifying audio packets, such as leveraging the UDP packet size without application or RTP header information [10], would help generalize this method to other VCAs.

7 CONCLUSION

Our goal has been to provide a framework for the study of interactivity QoE metrics in VCAs. We formally define a set of quantifiable metrics that correlate with VCA user interactivity experience. We then explore a spectrum of approaches for evaluating these metrics, focusing on the use of simulation as a scalable and generalizable approach. We create a validated state-machine simulation model of a VCA client behavior and demonstrate its capability in a variety of VCA use cases, as well as provide experimental insights into the effect of network latency and model parameters on interactivity metrics. We also show how these metrics can be measured with packet traces collected within real networks. As future work, we will perform further experimentation to generalize the collection of interactivity metrics using packet traces from a variety of VCAs, and better link the interactivity metrics to objective measures of QoE. We anticipate that the metrics, models, and methods presented in this paper will inform future work on understanding and measuring interactivity aspects of user QoE for video conferencing, as it becomes an increasingly popular form of communication.

REFERENCES

[1] Patrick Le Callet, Sebastian Moller, and Andrew Perkis. Qualinet White Paper on Definitions of Quality of Experience. *European Network on Quality of Experience in Multimedia Systems and Services (COST Action IC 1003)*, 2013.

[2] International Telecommunications Union. P.10: Vocabulary for performance, quality of service and quality of experience, 2017. URL https://www.itu.int/rec/T-REC-P.10-201711-I/en.

[3] Bart Jansen, Timothy Goodwin, Varun Gupta, Fernando Kuipers, and Gil Zussman. Performance Evaluation of WebRTC-Based Video Conferencing. *ACM SIGMETRICS Performance Evaluation Review*, 45(3):56–68, 2018.

[4] Hyunseok Chang, Matteo Varvello, Fang Hao, and Sarit Mukherjee. Can You See Me Now? A Measurement Study of Zoom, Webex, and Meet. In *Proc. ACM Internet Measurement Conference*, 2021.

[5] Rohan Kumar, Dhruv Nagpal, Vinayak Naik, and Dipanjan Chakraborty. Comparison of Popular Video Conferencing Apps Using Client-Side Measurements on Different Backhaul Networks. In *Proc. ACM Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, 2022.

[6] Insoo Lee, Jinsung Lee, Kyunghan Lee, Dirk Grunwald, and Sangtae Ha. Demystifying Commercial Video Conferencing Applications. In *Proc. ACM International Conference on Multimedia*, 2021.

[7] Kyle MacMillan, Tarun Mangla, James Saxon, and Nick Feamster. Measuring the Performance and Network Utilization of Popular Video Conferencing Applications. In *Proc. ACM Internet Measurement Conference*, 2021.

[8] Constantin Sander, Ike Kunze, Klaus Wehrle, and Jan Rüth. Video Conferencing and Flow-Rate Fairness: A First Look at Zoom and the Impact of Flow-Queuing AQM. In *Proc. Passive and Active Measurement (PAM)*, pages 3–19, 2021.

[9] Jia He, Mostafa Ammar, and Ellen Zegura. A Measurement-Derived Functional Model For the Interaction Between Congestion Control And QoE In Video Conferencing. In *Proc. Passive and Active Measurement*, 2023.

[10] Taveesh Sharma, Tarun Mangla, Arpit Gupta, Junchen Jiang, and Nick Feamster. Estimating WebRTC Video QoE Metrics Without Using Application Headers. In *Proc. ACM Internet Measurement Conference*, 2023.

[11] Jingxi Xu and Benjamin W. Wah. Exploiting Just-Noticeable Difference of Delays for Improving Quality of Experience in Video Conferencing. In *Proc. ACM Multimedia Systems Conference*, 2013.

[12] Jesse Frey, Jaden Pieper, and Tim Thompson. Mission Critical Voice QoE Mouth-to-Ear Latency Measurement Methods. *National Institute of Standards and Technology, NISTIR 8206*, 2018.

[13] E.A. Isaacs and J.C Tang. What video can and cannot do for collaboration: A case study. *Multimedia Systems*, 2:63–73, 1994.

[14] Abigail J. Sellen. Remote conversations: The effects of mediating talk with technology. *Human-Computer Interactions*, 10(4):401–444, 1995.

[15] Robert S. Fish, Robert E. Kraut, Robert W. Root, and Ronald E. Rice. Evaluating Video as a Technology for Informal Communication. In *Proc. ACM SIGCHI Conference on Human Factors in Computing Systems*, 1992.

[16] Andrew F. Monk and Caroline Gale. A Look Is Worth a Thousand Words: Full Gaze Awareness in Video-Mediated Conversation. *Discourse Processes*, 33(3): 257–278, 2002.

[17] David Grayson and Lynne Coventry. The Effects of Visual Proxemic Information in Video Mediated Communication. *ACM SIGCHI Bulletin*, 30(3):30–39, 1998.

[18] Rick van der Kleij, Jan Maarten Schraagen, Peter Werkhoven, and Carsten K. W. De Dreu. How Conversations Change Over Time in Face-to-Face and Video-Mediated Communication. *Small Group Research*, 40(4):355–381, 2009.

[19] Hiroaki Kawashima, Takeshi Nishikawa, and Takashi Matsuyama. Visual Filler: Facilitating Smooth Turn-Taking in Video Conferencing with Transmission Delay. In *Proc. ACM Extended Abstracts on Human Factors in Computing Systems*, 2008.

[20] Sebastian Egger, Raimund Schatz, and Stefan Scherer. It takes two to tango - assessing the impact of delay on conversational interactivity on perceived speech quality. In *Proc. Interspeech*, 2010.

[21] Catherine Lai, Jean Carletta, and Steve Renals. Modelling Participant Affect in Meetings with Turn-Taking Features. In *Proc. Workshop on Affective Social Speech Signals*, 2013.

[22] Marwin Schmitt, Simon Gunkel, Pablo Cesar, and Dick Bulterman. Asymmetric delay in video-mediated group discussions. In *International Workshop on Quality of Multimedia Experience (QoMEX)*, 2014.

[23] Katrin Schoenenberg, Alexander Raake, and Judith Koeppe. Why are you so slow? – Misattribution of transmission delay to attributes of the conversation partner at the far-end. *International Journal of Human-Computer Studies*, 72(5): 477–487, 2014.

[24] Marwin Schmitt, Judith Redi, Dick Bulterman, and Pablo S. Cesar. Towards Individual QoE for Multiparty Videoconferencing. *IEEE Transactions on Multimedia*, 20(7):1781–1795, 2018.

[25] Shrikant Garg, Ayushi Srivastava, Mashhuda Glencross, and Ojaswa Sharma. A Study of the Effects of Network Latency on Visual Task Performance in Video Conferencing. In *Proc. ACM Extended Abstracts on Human Factors in Computing Systems*, 2022.

[26] E. Geelhoed, A. Parker, D Williams, and M. Groen. Effects of Latency on Telepresence. *Technical Reports, HPL-2009-120, Hewlett-Packard Labs*, 2009.

[27] Momoko Nakatani, Yoko Ishii, Ai Nakane, Chihiro Takayama, and Fumiya Akasaka. Improving Satisfaction in Group Dialogue: A Comparative Study of Face-to-Face and Online Meetings. In *Proc. Human-Computer Interaction. Design and User Experience Case Studies*, 2021.

[28] Yang Xu, Chenguang Yu, Jingjiang Li, and Yong Liu. Video Telephony for End-Consumers: Measurement Study of Google+, IChat, and Skype. In *Proc. ACM Internet Measurement Conference*, 2012.

[29] Tarun Mangla, Emir Halepovic, Mostafa Ammar, and Ellen Zegura. eMIMIC: Estimating HTTP-Based Video QoE Metrics from Encrypted Network Traffic. In *2018 Network Traffic Measurement and Analysis Conference (TMA)*, 2018.

[30] Zoom. Zoom: Architected for Reliability, 2019. URL https://explore.zoom.us/docs/doc/Zoom_Global_Infrastructure.pdf.

[31] Zoom. Here's How Zoom Provides Industry-Leading Video Capacity, 2022. URL https://blog.zoom.us/zoom-can-provide-increase-industry-leading-video-capacity/.

[32] Mozilla. Signaling and video calling, 2023. URL https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API/Signaling_and_video_calling.

[33] Henning Schulzrinne, Stephen Casner, Ron Frederick, and Van Jacobson. RFC 3550: RTP: A Transport Protocol for Real-Time Applications, 2003. URL https://datatracker.ietf.org/doc/html/rfc3550.

[34] International Telecommunications Union. H.264: Advanced video coding for generic audiovisual services, 2016. URL https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-201602-S!!PDF-E&type=items.

[35] Google Developer. Core Technologies: VP9 Overview, 2023. URL https://developers.google.com/media/vp9.

[36] Xiph. Vorbis audio compression, 2016. URL https://xiph.org/vorbis/.

[37] Opus. Opus Interactive Audio Codec, 2023. URL https://opus-codec.org/.

[38] J. M. Valin, K. Vos, and T. Terriberry. RFC 6716: Definition of the Opus Audio Codec, 2012. URL https://datatracker.ietf.org/doc/html/rfc6716.

[39] W3C. Scalable Video Coding (SVC) Extension for WebRTC, 2023. URL https://www.w3.org/TR/webrtc-svc/.

[40] Chromium Source. Video coding in WebRTC, 2023. URL https://chromium.googlesource.com/external/webrtc/+/master/modules/video_coding/g3doc/index.md.

[41] Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. Overview of the Scalable Video Coding Extension of the H.264/AVC Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1103–1120, 2007.

[42] Peter Amon, Haoyu Li, Andreas Hutter, Daniele Renzi, and Stefano Battista. Scalable Video Coding and Transcoding. In *Proc. IEEE International Conference on Automation, Quality and Testing, Robotics*, 2008.

[43] Jitsi. Selective forwarding unit implementation of the jitsi videobridge, 2022. URL https://github.com/jitsi/jitsi-videobridge/blob/master/doc/sfu.md#bandwidth-estimations.

[44] Jitsi. Github repository, 2022. URL https://github.com/jitsi.

[45] M.J. Riley, I.E.G. Richardson, and B. Kohler. Low latency video communications over high bandwidth-delay networks using FEC. In *Proc. IEE Colloquium on Time Critical Data Communications*, 1994.

[46] Ralf M. Schreier and Albrecht Rothermel. A Latency Analysis on H.264 Video Transmission Systems. In *Proc. Digest of Technical Papers - International Conference on Consumer Electronics*, 2008.

[47] Mihir Mody, Pramod Swami, and Pavan Shastry. Ultra-low latency video codec for video conferencing. In *IEEE International Conference on Electronics, Computing and Communication Technologies*, 2014.

[48] Kevin M. McNeill, Mingkuan Liu, and Jeffrey J. Rodriguez. An Adaptive Jitter Buffer Play-Out Scheme to Improve VoIP Quality in Wireless Networks. In *Proc. IEEE Military Communications Conference*, 2006.

[49] Gaetano Carlucci, Luca de Cicco, Stefan Holmer, and Saverio Mascolo. Analysis and Design of the Google Congestion Control for Web Real-Time Communication (WebRTC). In *Proc. ACM International Conference on Multimedia Systems (MMSys)*, 2016.

[50] Paul T. Brady. Effects of transmission delay on conversational behavior on echo-free telephone circuits. *The Bell System Technical Journal*, 50(1):115–134, 1971.

[51] Brid O'Conaill, Steve Whittaker, and Sylvia Wilbur. Conversations over Video Conferences: An Evaluation of the Spoken Aspects of Video-Mediated Communication. *Human-Computer Interactions*, 8(4):389–428, 1993.

[52] S. Basu. A linked-HMM model for robust voicing and speech detection. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2003.

[53] Tanzeem Choudhury and Sumit Basu. Modeling Conversational Dynamics as a Mixed-Memory Markov Process. In *Proc. International Conference on Neural Information Processing Systems*, 2004.

[54] Karen Ruhleder and Brigitte Jordan. Co-Constructing Non-Mutual Realities: Delay-Generated Trouble in Distributed Interaction. *Computer Supported Cooperative Work*, 10:113–138, 2001.

[55] Lucas Seuren, Joseph Wherton, Trisha Greenhalgh, and Sara Shaw. Whose turn is it anyway? Latency and the organization of turn-taking in video-mediated

interaction. *Journal of Pragmatics*, 172:63–78, 2021.

[56] Lawrence K. Saul and Michael I. Jordan. Mixed Memory Markov Models: Decomposing Complex Stochastic Processes As Mixtures Of Simpler Ones. *Machine Learning*, 37:75–86, 1999.

[57] University of Edinburgh. AMI Corpus, 2006. URL https://groups.inf.ed.ac.uk/ami/corpus/.

[58] Emir Halepovic, Majid Ghaderi, and Carey Williamson. Multimedia application performance on a WiMAX network. In *Proc. SPIE Multimedia Computing and Networking*, 2009.

[59] Oliver Michel, Satadal Sengupta, Hyojoon Kim, Ravi Netravali, and Jennifer Rexford. Enabling Passive Measurement of Zoom Performance in Production Networks. In *Proc. ACM Internet Measurement Conference*, 2022.

[60] Institute of Electrical and Electronics Engineers. IEEE Standards for Local Area Networks: Carrier Sense Multiple Access With Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications. *ANSI/IEEE Std 802.3-1985*, 1985.

[61] Institute of Electrical and Electronics Engineers. IEEE Standard for Ethernet. *IEEE Std 802.3-2022 (Revision of IEEE Std 802.3-2018)*, 2022.

[62] Yih-Chiao Liu and G. Wise. Performance of a CSMA/CD Protocol for Local Area Networks. *IEEE Journal on Selected Areas in Communications*, 5(6):948–955, 1987.

[63] Chuan Heng Foh and M. Zukerman. Performance comparison of CSMA/RI and CSMA/CD with BEB. In *Proc. IEEE International Conference on Communications*, 2001.

[64] E. Wong and Chang-Joon Chae. CSMA/CD-based Ethernet over passive optical network for delivery of voice-over-IP traffic. In *Proc. Annual Meeting of the IEEE Lasers and Electro-Optics Society*, 2003.

[65] Konstantinos Voulgaris, Athanasios Gkelias, Imran Ashraf, Mischa Dohler, and A. H. Aghvami. Throughput Analysis of Wireless CSMA/CD for a Finite User Population. In *Proc. IEEE Vehicular Technology Conference*, 2006.

[66] Tao Ma, John Holden, and Rostislav Serota. Distribution of human response times. *Complexity*, 21:61–69, 2016.

[67] Daniel Zagar and Stephanie Mathey. When WORDS with Higher-frequency Neighbours Become Words with No Higher-frequency Neighbour (Or How to Undress the Neighbourhood Frequency Effect). *Reading as a Perceptual Process*, pages 23–46, 2000.

[68] Sandesh Dhawaskar Sathyanarayana, Kyunghan Lee, Dirk Grunwald, and Sangtae Ha. Converge: QoE-Driven Multipath Video Conferencing over WebRTC. In *Proc. ACM SIGCOMM Conference*, 2023.

[69] Raimund Schatz, Tobias Hoßfeld, and Pedro Casas. Passive YouTube QoE Monitoring for ISPs. In *Proc. International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, 2012.

[70] Vengatanathan Krishnamoorthi, Niklas Carlsson, Emir Halepovic, and Eric Petajan. BUFFEST: Predicting Buffer Conditions and Real-Time Requirements of HTTP(S) Adaptive Streaming Clients. In *Proc. ACM Conference on Multimedia Systems*, 2017.

[71] Matteo Varvello, Hyunseok Chang, and Yasir Zaki. Performance Characterization of Videoconferencing in the Wild. In *Proc. ACM Internet Measurement Conference*, 2022.