Generating Representative Video Teleconferencing Traffic

David DeAngelis
USC/ISI
deangeli@isi.edu

Alefiya Hussain
USC/ISI
hussain@isi.edu

Brian Kocoloski
USC/ISI
bkocolos@isi.edu

Calvin Ardi
USC/ISI
calvin@isi.edu

Stephen Schwab
USC/ISI
schwab@isi.edu

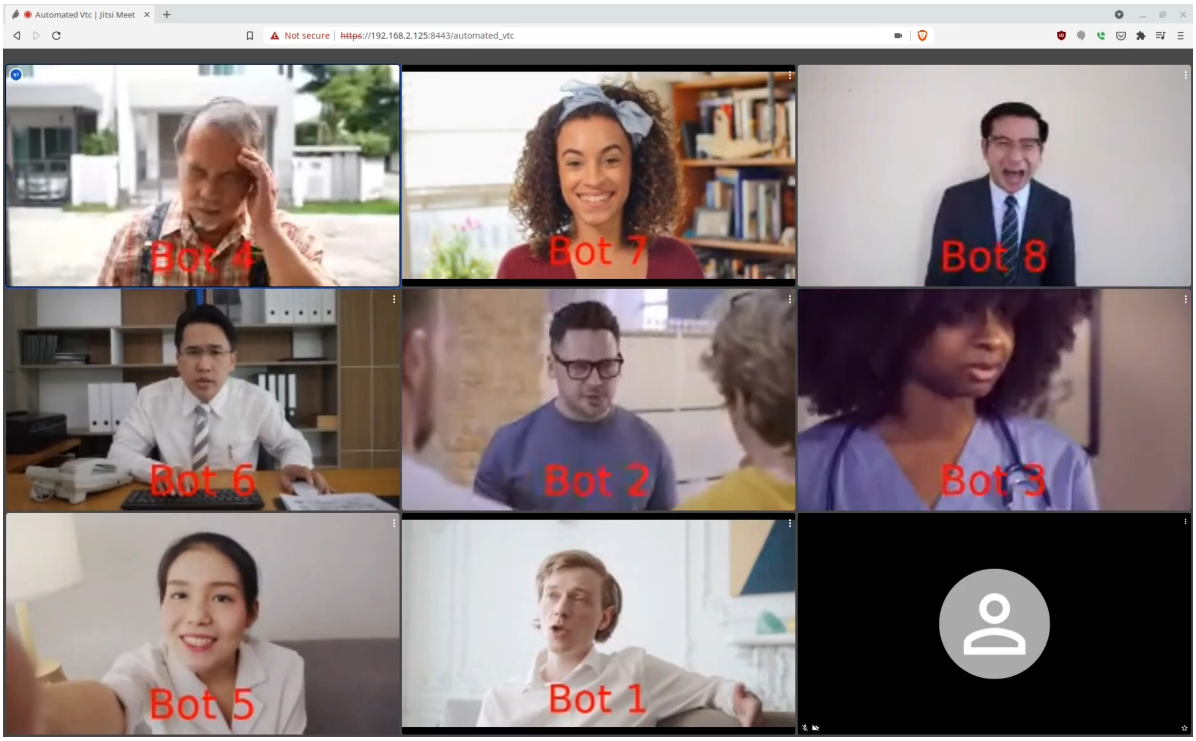


Figure 1: VTC traffic generator client screenshot

ABSTRACT

Video teleconferencing (VTC) is a dominant network application, yet there is a dearth of tools to generate such traffic for systematic and reproducible experimentation. We present a framework to create representative video teleconferencing traffic and discuss our methodology for behavioral control of multiple bots to create human-like dialog coordination, including interactive talking and silence patterns. Our framework can be coupled with proprietary commercial VTC applications as well as deployed completely within a testbed environment to benchmark emerging networking technology and evaluate the next generation of traffic classification, quality of service (QoS) algorithms, and traffic engineering systems. Our traffic generators are open source and freely available at https://mergetb.org/projects/searchlight/.

CCS CONCEPTS

• **Networks** → **Network performance evaluation**; • **Human-centered computing** → *Interactive systems and tools*; *User models*.

KEYWORDS

network traffic generation, video teleconference, VoIP, cybersecurity testbeds

ACM Reference Format:
David DeAngelis, Alefiya Hussain, Brian Kocoloski, Calvin Ardi, and Stephen Schwab. 2022. Generating Representative Video Teleconferencing Traffic. In *Cyber Security Experimentation and Test Workshop (CSET 2022), August 8, 2022, Virtual, CA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3546096.3546107

CSET 2022, August 8, 2022, Virtual, CA, USA

ACM ISBN 978-1-4503-9684-4/22/08…$15.00
https://doi.org/10.1145/3546096.3546107

1 INTRODUCTION

Large-scale video teleconference (VTC) environments are commonplace today as more people are working from home and participating in meetings and conferences virtually. However, there are limited tools to model human-like VTC traffic for *systematic and disciplined* experimentation. Prior work in traffic generation and simulation typically focus on maximizing throughput for benchmarks and stress testing [8] or on simulating the behavior of the underlying protocols (TCP, UDP, and others) [2, 12, 16].

Networking and cyber security researchers need VTC traffic generation tools in order to develop and evaluate the next generation of systems, including those for network traffic classification and quality of service management. These tools allow experimenters to test and evaluate their solutions during development, and include reconfigurable client and server components.

Additionally, to create *representative* traffic on-the-wire, the interaction between the various participants in the VTC session should be *human-like*. In section 2, we present the behavior control model used for human-like interaction with the bots. While preliminary at this stage, this model enables *reproducible* VTC experiments by emulating the process of "talking" and "listening" with a systematic and well-defined methodology.

In section 3 we discuss the various components in the VTC framework and how they can be configured for a networking experiment. Typically, a networking experiment involves introducing some network dynamics, such as rate limiting and flow rerouting, that will have an impact on the VTC experience. In section 4, we discuss how our framework collects Quality of Experience (QoE) measurements to characterize the impact of the network on the end user. In section 5.1, we discuss an end-to-end case study of how our VTC traffic generator can be used for an evaluation on a emulation-based testbed. Finally, in section 6, we outline future directions to expand this initial framework.

2 BEHAVIOR MODELING

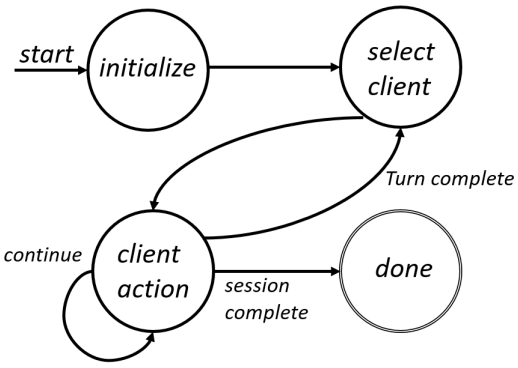


Figure 2: Finite state machine model of a multi-party discourse controller

A primary component of a realistic VTC traffic generation system is a mechanism for creating human-like behavior. In the case of VTC, these behaviors include identifying a speaker, switching speakers to match realistic dialog patterns, producing a variety of dialog clips in a set of unique voices, realistic pauses, and more. This generative approach is necessary for rigorous experimentation and is superior to simple VTC traffic playback because complex networks utilize sophisticated compression and caching techniques that could capture the repetition of traffic replays. Beyond VTC, the technique proposed here generalizes to any multi-party discourse. A portion of this mechanism is represented formally in Figure 2. The VTC controller described in section 3.2 implements this finite state machine to orchestrate dialog.

3 VTC FRAMEWORK

The VTC traffic generator implements the behavioral control model described in section 2. The VTC framework is built from a collection of separate technologies which serve as building blocks for automated traffic generation in a testbed environment. This section describes how each subcomponent in this framework operates. Figure 3 shows the overall architecture of the VTC framework, with VTC clients, a VTC controller, and a VTC application running on separate machines of an experiment testbed.

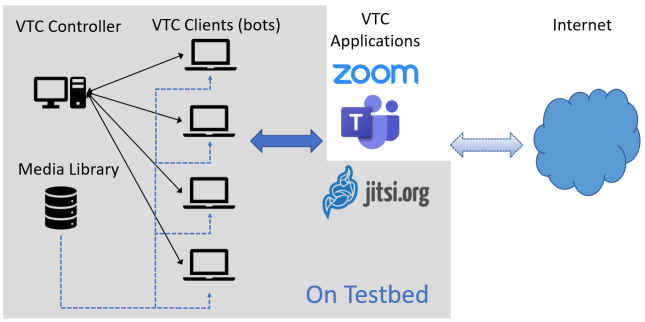


Figure 3: VTC traffic generator system architecture

3.1 VTC Application

The VTC application is a 3rd party internet based teleconference application such as Zoom, Microsoft Teams, Google Hangouts, or a self-hosted application such as Jitsi Meet (see Figure 1). The traffic generator presented here can operate on any VTC application, but we have chosen to focus on a dockerized self-hosted Jitsi instance within our testbed [15] for experiment repeatability. With the exception of 2-party VTC sessions that are directly peer-to-peer, all audio and video feeds are sent to and received from the VTC application.

3.2 VTC Controller

The VTC controller is the component of the framework that serves as the brain of the session. This component uses a behavioral model (see section 2) that initiates clients, connects clients to the VTC session, orchestrates dialog, and closes the VTC session. The dialog orchestration is currently implemented as a loop that selects a client bot, where all bots except the currently speaking one are equally probable. Next, the selected bot is instructed to take a dialog turn, while the controller waits until the dialog is complete, then the controller selects a different client to speak. The controller directs the clients using the XML-RPC protocol on the control plane.

3.3 VTC Clients

The VTC clients are implemented as XML-RPC based servers that run continuously, awaiting instruction from the VTC controller. When they connect to the VTC session their assigned video track will play on loop into a virtual camera device. When instructed to speak, they select a conversation at random from a set of pre-generated dialog tracks in their assigned voice, and they speak for a variable number of sentences that is governed by a normal distribution $\lfloor |\mathcal{N}(0, 2)| \rceil + 1$.

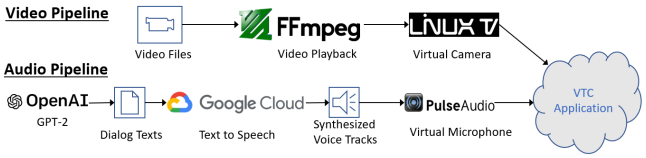


Figure 4: Video and audio media pipelines

3.3.1 Virtual Devices. The VTC clients contain virtual camera and microphone devices that connect the bots to the VTC application. Mimicking the physical world with virtual devices allows us to use the standard input interfaces that the VTC applications provide. Figure 4 shows the video and audio pipelines from origination through plumbing the virtual devices into the VTC application. The virtual camera is a V4L2 loopback kernel module [14] that is fed with video files played back using FFmpeg. The appropriate bot name is superimposed onto the video feed to assist in development and clearly indicate the presence of a bot in the VTC session, see Figure 1. The virtual microphone is created using the Pulse Audio sound server to play a dialog track into a virtual speaker, where the speaker output is connected to the input of a software microphone, which is selected by the VTC application using browser automation.

These virtual devices present to the VTC application in an identical fashion to physical devices. Via the V4L2 loopback kernel module, the virtual camera appears as a physical camera, and the video media files are preprocessed to match popular camera output resolutions and frame rates. Dialog audio tracks are generated in raw WAV format, compressed using lossless FLAC encoding, and fed directly into the virtual microphone. This ensures that no audio compression artifacts are present other than those attributable to the VTC application itself. However, adding ambient and environmental noise would add further to the realism.

3.4 Media Library

Figure 3 shows a shared media library that is fed into the virtual devices, comprising the audio and video content of the VTC session. This is generated *a priori* using a process outlined in Figure 4. The video files are currently sourced online from websites with free-use licenses. The audio dialog tracks are procedurally generated in a 3 step process:

(1) OpenAI's GPT-2 language model generates dialog texts [10].
(2) Manual curation removes inappropriate content.
(3) Google Cloud Text-to-Speech service is employed to generate FLAC audio tracks using a variety of realistic WaveNet voices [5].

3.5 Automation

Lastly, several layers of automation are employed to ensure reliable, repeatable experimentation with the VTC traffic generator on a networked testbed. First, on each VTC client the Microsoft Playwright web application testing framework provides an automated web browser that is scripted to join/exit a VTC session and select the virtual input devices. This browser automation is specific to the self-hosted Jitsi Meet VTC application. Applying this technology to a different platform requires adapting the browser automation for the new platform's web client. These web clients for 3rd party VTC platforms may not have a stable user interface, thereby introducing risk. Beyond client automation, the Ansible automation platform is used to configure each component shown in Figure 3, including setting experimental parameters, launching experiments, and collecting results.

4 QUALITY OF EXPERIENCE (QOE)

As described in section 3, effort was taken to ensure the variety, variability, and fidelity of the VTC client media that is fed into the VTC application. However, network conditions, whether they are accidental or intentional, transient or permanent, can adversely affect the Quality of Experience (QoE) of the VTC application. QoE is an important measure that is used to characterize the impact of network and device conditions on the end user. However, QoE measures for VTC are constantly evolving [11][3].

A VTC session is comprised of synchronized, bidirectional video and audio feeds to each participant. The gold standard for QoE for video and audio streaming is a subjective evaluation called the mean opinion score (MOS)[6], which is essentially an ordinal scale of perceived quality. Calculating MOS is difficult because it requires human evaluation and is outside the scope of this work, so we turn to automated measures for QoE. Because VoIP is analogous to VTC, only without video, we argue that VoIP measures of QoE are a starting point for evaluating VTC QoE, since in many cases a properly functioning, low latency audio track is essential to a productive VTC session, particularly when multiple participants must take turns speaking. The video track is often of secondary importance, though we did informally observe video quality degradation as the link capacity was manipulated during VTC sessions.

Two of the most critical and readily obtained measures of QoE for VTC are latency, measured in round trip time (RTT), and jitter, which can be described as temporal deviations from the input signal. Our VTC traffic generator can drive a self-hosted instance of the Jitsi VTC application, and a core component called the Jitsi Videobridge publishes session statistics that can serve as an indicator of QoE including latency, jitter, and more [7]. Beyond these initial measures, we plan to incorporate video-specific metrics defined in the Common Media Client Data (CMCD) specification [4] and accessible through the WebRTC stats module [1], including encoding bitrate and buffer starvation as measures of video QoE.

5 EVALUATION

The initial evaluation presented here has two parts: a case study and an analysis of network traffic generated by the VTC generator. The case study to demonstrates a deployment scenario of the VTC

traffic generator and explores its performance under varying network conditions. The traffic analysis section compares the synthetic network traffic generated by the VTC traffic generator to network traffic generated by a live human VTC session.

5.1 Case Study

Figure 5 shows an experimental topology on a cyber experimentation testbed [15] where $h0$ hosts the Jitsi application and $h2$-$h11$ each host one of 10 VTC clients feeding 1080p video. Initially each link has an available bandwidth of 50 Mbps, full duplex, and at $t = 175s$ the link between $c0$ and $c1$ is dropped to 10 Mbps, full duplex.

Each client node consists of 8 virtualized CPU cores with 16 GB of RAM. The least powerful nodes we have deployed to contain a quad core Intel Atom CPU and 2GB of RAM. On these nodes the automated Chromium browser is CPU limited when 1080p 60fps video streams are enabled. Low resolution streams alleviate this limitation, as does provisioning more powerful experiment nodes.

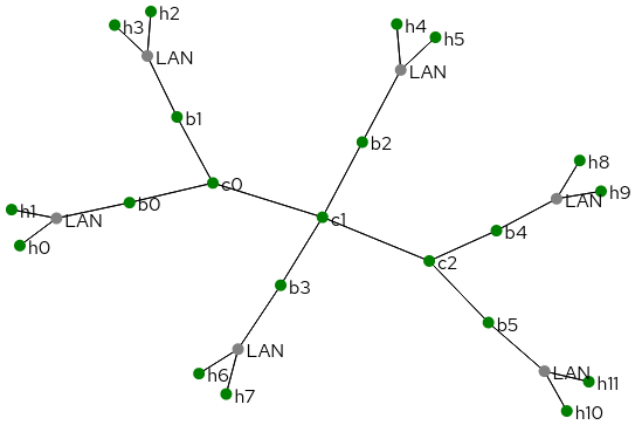
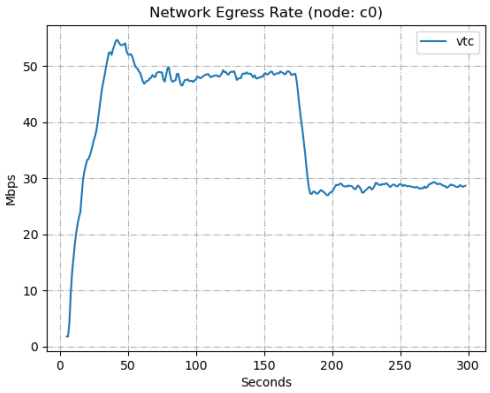


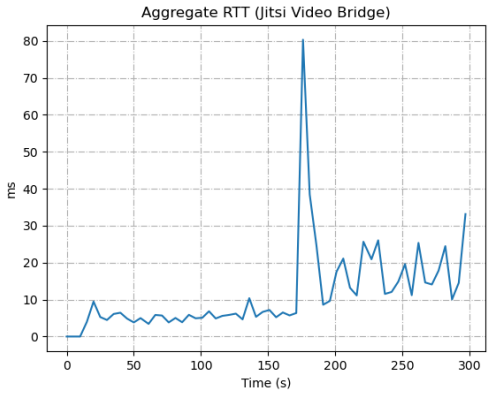
Figure 5: Experiment network topology

Figure 6a shows the bandwidth of the VTC session, measured as the egress rate on core router $c0$. At $t = 175s$ when we deliberately limit the bandwidth between $c0$ and $c1$ dropping the link capacity from 50 Mbps to 10 Mbps. We observe that the VTC session bandwidth on node $c0$, calculated as the sum of the VTC egress traffic on the 3 $c0$ links drops from from roughly 50 Mbps to under 30 Mbps. Simultaneously, figure 6b shows a significant spike in latency, but latency quickly recovers to tolerable levels due to the underlying WebRTC protocol that powers the Jitsi VTC application. This also coincides with an observed degradation in quality of the video feeds originating on the clients that depend on the degraded, saturated link.

This case study demonstrates that the VTC traffic generator can be deployed on realistic network topologies, has significant requirements for network resources, and responds dynamically to changing network conditions. The study presented here is just a single demonstration deployment, but Ansible automation enables rapid, repeatable deployments and data collection for testbed or locally hosted experiments.



(a) VTC session bandwidth, measured as egress from $c0$



(b) Aggregrate latency as reported by the Jitsi Videobridge

Figure 6: VTC session network characteristics. Note: link capacity experimental variable changed at $t = 175s$.

5.2 Traffic Analysis

This initial traffic analysis compares synthetic VTC traffic to live VTC traffic. The live capture was taken by conducting a 5 minute 3-way conversation between human participants using a self hosted Jitsi Meet server, with the server and all clients on the corporate network. The network interface on the Jitsi Meet host was tapped to ensure the traffic between all clients and the Jitsi application was captured.

The synthetic VTC traffic was captured on the Lighthouse [15] testbed by tapping the same Jitsi Meet network interface. Two 5 minute sessions were captured, one with 270p video resolution from all clients, and another with 1080p resolution video. Table 1 shows summary statistics of the packet captures along with the traffic flows between each end client {A,B,C} and the Jitsi host {H}.

The first row represents a live human VTC session, whereas rows 2 and 3 represent synthetic traffic created by the VTC generator. Strong similarities in the packet captures across all three runs are apparent, but there are some notable differences, particularly with the packet capture size and with the UDP flow size. We hypothesize that some of the discrepancy in capture sizes is due to

Table 1: Network traffic comparison between bots and human operators

clients	source resolution	size (MB)	duration (m:s)	protocol	UDP flow size (MB)					
					A→H	B→H	C→H	H→A	H→B	H→C
3x human	varies	479	5:06	UDP	135	133	133	19	19	17
3x bot	270p	171	5:16	UDP	38	37	37	15	16	16
3x bot	1080p	373	5:09	UDP	51	130	126	15	16	15

a combination of 2 factors. First, there is currently no ambient or environmental noise in addition to the dialog in bot sessions. In a live conversation with unmuted participants there is a constant audio feed flowing from the client to the host. Secondly, the WebRTC technology underlying Jitsi uses AVC and VP8 video compression. We suspect that the looping videos might be more compressible than a live video feed. The relative UDP flow sizes between clients and the host is consistent across all runs, though we observed larger upload flows than download flows, likely due to the videobridge forwarding lower bandwidth streams. This initial analysis examines flow volume, but not conversation dynamics, inviting further investigation into the temporal aspects of both live and synthetic VTC sessions.

6 FUTURE WORK AND CONCLUSION

This paper presented a new tool for creating realistic video teleconference and voice over IP traffic on a network. This tool has been used at modest scale on several cybersecurity testbeds. Along with other traffic generators, this work has already played an instrumental role in developing new networking technologies for managing Quality of Service for distributed applications on the internet [9]. As hosted VTC applications continue to gain favor over older point to point VoIP protocols, technologies to measure, manage, and secure networks will continue to benefit from realistic VTC traffic generation as presented here.

There are several compelling areas to improve the fidelity and functionality of this VTC traffic generator. Most importantly, conversation dynamics are governed by a heuristic that is not grounded in empirical research. Studying the dynamics of recorded conversations, specifically the temporal distributions of speaker participation, would add fidelity to the traffic generation and perhaps lend insight into the social sciences too.

Currently video media is sourced externally and cultivating a large video library to ensure variety in traffic is cumbersome. Nascent techniques involving generative adversarial networks have the potential to automatically create ultra realistic video feeds [13] without looping. In this vein, audio and video media is currently generated *a priori*, but on-the-fly media generation would represent an evolutionary step in VTC traffic generation.

Lastly, adapting the VTC traffic generator to host sessions involving a combination of automated bots and live humans could help advance conversational AI technologies.

The VTC/VoIP traffic generator presented here, along with additional tools and traffic generators for network experimentation, are available to the public at: https://mergetb.org/projects/searchlight/.

ACKNOWLEDGMENTS

This work was sponsored by Sandia National Laboratory (SNL) under PO2160586. This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

REFERENCES

[1] Harald Alvestrand, Varun Singh, and Henrik Boström. 2022. *Identifiers for WebRTC's Statistics API.* Retrieved July 6, 2022 from https://w3c.github.io/webrtc-stats/

[2] Doreid Ammar, Thomas Begin, and Isabelle Guerin-Lassous. 2011. A New Tool for Generating Realistic Internet Traffic in NS-3. In *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques* (Barcelona, Spain) *(SIMUTools '11)*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Brussels, BEL, 81–83.

[3] Doreid Ammar, Katrien De Moor, Min Xie, Markus Fiedler, and Poul Heegaard. 2016. Video QoE killer and performance statistics in WebRTC-based video communication. In *2016 IEEE Sixth International Conference on Communications and Electronics (ICCE)*. 429–436. https://doi.org/10.1109/CCE.2016.7562675

[4] Abdelhak Bentaleb, May Lim, Mehmet N. Akcay, Ali C. Begen, and Roger Zimmermann. 2021. Common Media Client Data (CMCD): Initial Findings. In *Proceedings of the 31st ACM Workshop on Network and Operating Systems Support for Digital Audio and Video* (Istanbul, Turkey) *(NOSSDAV '21)*. Association for Computing Machinery, New York, NY, USA, 25–33. https://doi.org/10.1145/3458306.3461444

[5] Google. 2022. *Cloud Text-to-Speech.* Retrieved May 13, 2022 from https://cloud.google.com/text-to-speech

[6] Amendment Itu-T. 2006. 1: Recommendation P. 10/G. 100. *New Appendix I–Definition of Quality of Experience (QoE), Telecommunication Standardization Sector of Itu* 100 (2006), 2007.

[7] Jitsi. 2022. *Jitsi Videobridge Server Statistics.* Retrieved May 13, 2022 from https://github.com/jitsi/jitsi-videobridge/blob/master/doc/statistics.md

[8] ESnet / Lawrence Berkeley National Laboratory. 2020. iperf3 v3.9. https://software.es.net/iperf/

[9] John-Francis Mergen. 2019. *DARPA Searchlight.* Defense Advanced Research Projects Agency. Retrieved May 13, 2022 from https://www.darpa.mil/program/searchlight

[10] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[11] Marwin Schmitt, Judith Redi, Dick Bulterman, and Pablo S. Cesar. 2018. Towards Individual QoE for Multiparty Videoconferencing. *IEEE Transactions on Multimedia* 20, 7 (2018), 1781–1795. https://doi.org/10.1109/TMM.2017.2777466

[12] Joel Sommers, Hyungsuk Kim, and Paul Barford. 2004. Harpoon: A Flow-Level Traffic Generator for Router and Network Tests. *SIGMETRICS Perform. Eval. Rev.* 32, 1 (June 2004), 392. https://doi.org/10.1145/1012888.1005733

[13] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. 2018. MoCoGAN: Decomposing Motion and Content for Video Generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[14] umlaeute. 2022. *v4l2loopback - a kernel module to create V4L2 loopback devices.* Retrieved May 13, 2022 from https://github.com/umlaeute/v4l2loopback

[15] USC/ISI. 2022. *The Lighthouse testbed.* University of Southern California, Information Sciences Institute. Retrieved May 13, 2022 from https://gitlab.com/mergetb/facilities/lighthouse

[16] Michele C. Weigle, Prashanth Adurthi, Félix Hernández-Campos, Kevin Jeffay, and F. Donelson Smith. 2006. Tmix: A Tool for Generating Realistic TCP Application Workloads in Ns-2. *SIGCOMM Comput. Commun. Rev.* 36, 3 (July 2006), 65–76. https://doi.org/10.1145/1140086.1140094