RTC Relay Server Infrastructure Study

Peiqing Chen
University of Maryland
College Park, Maryland, USA
pqchen99@umd.edu

Sujal Umrikar
University of Maryland
College Park, Maryland, USA
sumrikar@umd.edu

Guowu Xie
Meta Platform Inc.
Menlo Park, California, USA
woo@meta.com

Zaoxing Liu
University of Maryland
College Park, Maryland, USA
zaoxing@umd.edu

ABSTRACT

Real-Time Communication (RTC) relay servers are critical for ensuring connectivity when peer-to-peer (P2P) links fail, directly impacting end-to-end latency, reliability, and user quality of experience (QoE). However, commercial RTC platforms keep their relay infrastructure—including deployment locations, targeting logic, and fallback mechanisms—opaque, leaving a gap in our understanding of real-world RTC system design. In this paper, we plot a global relay map for three major RTC applications (Zoom Free/Business, WhatsApp, and Discord) by automating cross-region calls over 13 Azure cloud regions worldwide. From our observations, a denser deployment of relay nodes and a better targeting strategy give WhatsApp on average 69 ms, 31 ms, and 35 ms lower end-to-end latency per call compared with Zoom Free, Zoom Business, and Discord, respectively. These insights provide valuable input for next-generation relay infrastructure design.

ACM Reference Format:
Peiqing Chen, Sujal Umrikar, Guowu Xie, and Zaoxing Liu. 2026. RTC Relay Server Infrastructure Study. In *ACM/IRTF Applied Networking Research Workshop (ANRW '26), July 20, 2026, Vienna, Austria.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3822163.3827928

1 INTRODUCTION

Real-time communication (RTC) is a cornerstone technology that connects people through interactive audio, video, and screen-sharing, and it underpins everyday services such as Zoom, WhatsApp, and Discord [3, 9, 12, 20]. RTC calls require establishing a stable, low-latency data transmission channel between a caller and a callee. When a direct peer-to-peer (P2P [16, 27]) path cannot be established—e.g., due to NATs [22, 24] or firewalls [5, 25]—the call falls back to relayed transmission via a TURN server [23] (a.k.a., relay server), i.e., every media packet is first sent to a relay which then forwards it to the other endpoint. Relay-mode transmission therefore fundamentally changes the data path and the performance semantics of a call [19, 23, 26].

In this paper, we study how the relay server infrastructure affects RTC call performance and reliability from the perspective of a service provider. A fundamental design decision is the construction of the **relay map**. Given a fixed budget—i.e., a fixed number of servers that can be deployed or rented—providers must decide how to place relay servers globally. One end of the design space favors a large number of geographically distributed points of presence (PoPs), which increases the likelihood that a call can be served by a nearby relay and thus reduces latency, but at the cost of higher operation and maintenance overhead and more complex load balancing. The opposite end favors a small number of centralized locations, which simplifies maintenance and enables coarse-grained load pooling, but often incurs higher latency for a large fraction of users. Beyond relay placement, providers must determine the **relay targeting** strategy—how a specific relay is selected for a given call when multiple candidates are available. Finally, robust **relay fallback** mechanisms are critical for maintaining call reliability when relays or transport paths fail, such as due to server outages or regional network disruptions. Together, relay mapping, targeting, and fallback decisions directly shape call latency, loss characteristics, and end-to-end reliability.

Measuring relay infrastructures and their effect on calls poses two practical challenges. First, we must exercise calls from geographically diverse origins to observe the relay locations globally. Second, we must generate a sufficiently large number of controlled calls so that targeting and fallback behaviors become visible and statistically meaningful. Addressing both requirements simultaneously while using real mobile clients is non-trivial.

To our knowledge, there is no prior cross-platform, global-scale measurement that maps and compares relay infrastructures of popular RTC applications; existing work focuses on protocol compliance, P2P performance, or measurements of individual applications [8, 12, 15, 20]. In this paper, **we present the first measurement and comparison study of commercial RTC relay infrastructures across major applications**. Specifically, we study three widely used RTC platforms—Zoom, Discord, and WhatsApp—further distinguishing between Zoom's Free and Business account tiers. To overcome the above challenges, we (1) connect physical mobile clients to WireGuard VPN endpoints in a set of global cloud regions, (2) automate large numbers of one-on-one (1-on-1) calls using Xcode-based device control to sweep caller–callee region pairs, and (3) capture packet traces at the cloud proxies to identify relay IPs and infer relay selection and fallback behavior.

ANRW '26, July 20, 2026, Vienna, Austria

ACM ISBN /2026/07
https://doi.org/10.1145/3822163.3827928

(a) Zoom Free (b) Zoom Business (c) Discord (d) WhatsApp

Figure 1: Relay maps for three representative RTC applications. Each red dot denotes a geographic location where at least one relay server IP was detected.

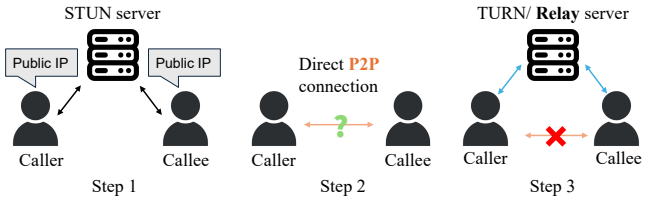

Figure 2: Connection establishment in a 1-on-1 call.

Summary of findings. Using this framework, we uncover three patterns across applications. **(i) Relay map density varies significantly** (Figure 1): Zoom Free uses only two relay locations, while Zoom Business and Discord are globally distributed and WhatsApp has the densest footprint. **(ii) Relay targeting policies differ** in how caller and callee locations influence selection: WhatsApp uses latency-aware multi-probing to pick the nearest responsive relay, whereas Zoom and Discord assign relays based on caller location. **(iii) Fallback robustness varies widely**: WhatsApp opens 3 UDP and 3 TCP fallback sessions toward 3 different relays, while Discord has no fallback beyond a single UDP session.

Measured impact on call latency. A denser relay deployment and better targeting shorten call latency. On intra-continent pairs, WhatsApp is on average 144/27/24 ms faster than Zoom Free/Zoom Business/Discord; on inter-continent pairs, WhatsApp is 47/32/39 ms faster.

2 BACKGROUND AND MOTIVATION

RTC applications generally support two call modes: 1-on-1 calls and group calls with multiple participants. In this paper, we focus on 1-on-1 calls for two reasons. First, 1-on-1 calls dominate real-world RTC usage, accounting for over 80% of calls in practice since 2020 [4, 14]. Second, group-call infrastructures are inherently more complex: beyond relay servers, their data transmission paths are also shaped by additional components such as Selective Forwarding Units (SFUs) [30], making it difficult to isolate and quantify the impact of relay infrastructure on call performance through relay-focused measurements alone. Accordingly, we restrict our scope to 1-on-1 calls and defer the study of group-call infrastructures to future work. In this paper, we use the term **relay server** to refer specifically to **TURN servers** that forward media traffic between two client devices.

2.1 How Relay Works in a 1-on-1 Call

Figure 2 illustrates the typical connection establishment of a 1-on-1 call, which sets up a channel for time-sensitive media (e.g., RTP [26]/RTCP [13]) between the two endpoints.

Step 1: Address discovery via STUN. Each client queries a STUN server to learn its *server-reflexive candidate*—the public-facing IP/port assigned by its NAT.

Step 2: Direct P2P connectivity check. The clients run STUN binding checks; if a pair is reachable, Interactive Connectivity Establishment (ICE) [15] selects it and media flows directly over a peer-to-peer (P2P [16]) path with minimal latency.[1]

Step 3: Relay-based fallback using TURN. When direct connectivity fails—e.g., due to symmetric NATs [22, 24], UDP-blocking firewalls [5, 25], or enterprise network policies [11]—an endpoint requests an allocation from a TURN server, which assigns a relayed transport address and forwards media on its behalf. All media packets then traverse the client–relay–client path, so call latency and loss are determined by both legs and the application's relay selection.

2.2 Measurement Framework Overview

Measuring relay infrastructures in commercial RTC applications is challenging for two reasons. First, relay deployment and selection logic are proprietary and closed-source, and applications do not expose internal relay maps or decision processes. Second, relay behavior is highly dynamic: the relay used for a given call may vary with caller and callee locations, call timing, and account type. We therefore adopt a third-party, black-box measurement approach that infers relay infrastructure properties solely from network traffic generated by controlled end-host call experiments.

Application selection. We study three popular RTC applications: Zoom, Discord, and WhatsApp, focusing on their mobile clients and 1-on-1 calls. These applications represent different segments of the RTC ecosystem and have large, diverse user bases. Zoom is the leading dedicated video-conferencing platform, WhatsApp is among the most widely used messaging applications globally with deeply integrated voice and video calling, and Discord is the dominant platform for real-time voice and text communication in gaming and online communities [14, 21, 29]. Together, they span a wide spectrum of relay deployment strategies—from minimal

[1] Some RTC applications (e.g., Discord) do not attempt P2P connectivity and always use relay-based media delivery.

centralized infrastructure (Zoom Free) to moderate global distribution (Zoom Business, Discord) and dense worldwide deployment (WhatsApp)—making them representative case studies for comparing relay infrastructure design choices. For Zoom, we explicitly distinguish between *Free* [33] and *Business* [31] account tiers for two reasons. [2] First, Zoom's public documentation indicates that paid accounts can opt in or out of specific data-center regions, while Free accounts are restricted to a provisioned region [34]. Second, although Zoom publishes IP address ranges used by its services [32], these disclosures do not specify which IPs correspond to TURN or relay servers, nor their roles in media forwarding, making direct measurement necessary to identify relay infrastructure differences across account tiers.

Call generation and traffic collection. To enable large-scale and geographically diverse measurements, we take two steps: 1) We connect physical iPhones to virtual machines deployed in multiple Azure regions using WireGuard VPN tunnels, causing calls to appear as originating from different geographic locations (see §3 for details); and 2) We use Xcode-based automation scripts to repeatedly generate 1-on-1 calls across different caller–callee region pairs. All network traffic is captured at the cloud VMs for subsequent analysis.

Relay inference. We analyze the collected network traces to identify relay servers and characterize their behavior. Relay IPs are extracted by isolating RTP flows associated with relay-mode calls [8]. To infer each relay IP's geographic location and the organization operating it, we query IP intelligence services (e.g., ipinfo) [2]. To validate the accuracy of these inferences, we additionally run traceroute and active latency measurements toward the inferred relay IPs and check for consistency between the measured path/RTT characteristics and the reported locations. Together, these techniques enable us to reconstruct each application's relay map, relay targeting policies, and fallback behavior from end-host observations.

2.3 Related Work

Prior work on RTC relays falls into three lines. (i) Measurement studies of individual applications—Zoom, Skype, Google Meet, Webex—characterize call latency, availability, and media behavior under varying networks [6, 7, 18, 28], but treat relaying only as an implicit fallback and do not reconstruct relay maps, targeting, or fallback strategies. (ii) Aggregate measurements quantify how often RTC systems use TURN in the wild [22], without distinguishing between applications or providers. (iii) Systems work optimizes the TURN relay itself for forwarding efficiency and scalability [17], operating at the server-design level rather than on how commercial RTC services actually deploy, select, and orchestrate their relay fleets.

In contrast to these lines of work, no prior study provides a systematic, cross-application measurement of commercial RTC relay infrastructures, nor compares relay maps, relay targeting policies, and fallback mechanisms across major platforms and quantifies their impact on end-to-end call performance and reliability.

Azure node	City / Proximity	Global Region
East US 2	Boydton, Virginia, USA	America
Central US	Des Moines Iowa, USA	America
West US	San Francisco, California, USA	America
South Central US	San Antonio, Texas, USA	America
Chile Central	San Diego, Chile	South America
UK South	London, United Kingdom	Europe
Poland Central	Warsaw, Poland	Europe
UAE North	Abu Dhabi, UAE	Middle East
Japan East	Tokyo (Saitama), Japan	Asia Pacific
Central India	Pune, India	Asia Pacific
South Africa North	Johannesburg, South Africa	Africa
Australia East	Sydney (New South Wales), Australia	Asia Pacific
Malaysia West	Kuala Lumpur, Malaysia	Asia Pacific

Table 1: Selected Azure VM regions, their proximity cities, and Azure global region categories.

3 MEASUREMENT METHODOLOGY

Our objective is to discover as many relay server locations of RTC applications worldwide as possible. Achieving this requires initiating RTC calls from diverse geographic regions. However, it is most challenging to deploy mobile devices across the globe physically. To overcome this, we leverage Azure VMs distributed worldwide to serve as VPN proxies. Each VM hosts a VPN server, and our test devices connect to it via WireGuard [10], forcing all mobile traffic through the assigned VM. This setup allows us to emulate RTC calls originating from the VM's regions. Meanwhile, we block the P2P connection between the VMs to avoid forming P2P transmission during the calls. Finally, we capture packets at the VM using Wireshark and analyze the RTP packets exchanged with relay servers to identify their IP addresses and geographic locations.

3.1 Call Experiment Setup

3.1.1 Device and Network Setup. We selected two iPhone 11 devices as the caller and the callee. Each phone connects to an Azure VM through a WireGuard VPN tunnel, configured to forward all mobile traffic through the chosen VM. A controller MacBook laptop orchestrates the experiments. It remotely manages the Azure VMs, starting and stopping traffic captures, issuing commands to block specific IPs, and ensuring synchronization between devices.

3.1.2 Caller and Callee Location Setup. To best emulate caller and callee locations worldwide under the time and resource constraints, we select a representative set of Azure VM regions across different continents, as listed in Table 1. We iterate the caller and callee through each VM, respectively, forming 169 different pairs. Each caller–callee pair is tested by initiating the same call 10 times consecutively. This repetition is necessary because applications do not always select the same relay server across different call attempts. Multiple repetitions provide richer observations and improve the reliability of our measurement results.

3.1.3 Experiment Procedure and Trace Capture. Before each experiment, both phones terminate background applications, connect to

[2] Zoom also has a school/education account. We do not include that due to access difficulty.

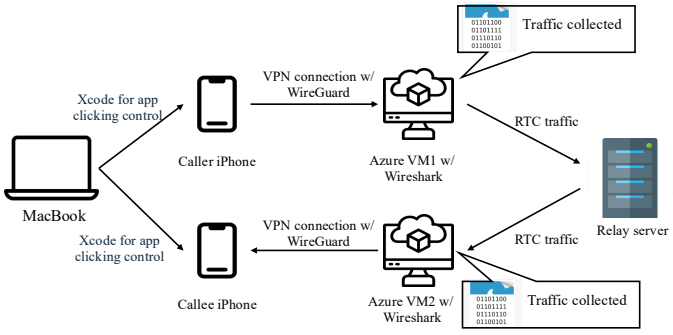

Figure 3: Relay server measurement methodology. Using 2 VMs from Microsoft Azure to mimic the caller and callee location at different places around the globe.

WiFi, and route all traffic through assigned Azure VMs via WireGuard.[3] We implement an automation program on a MacBook using Xcode to control the iPhones, which automatically restarts the RTC application, initiates a call, and maintains the call for one minute with both the microphone and the camera on. The Wireshark trace-capturing commands are also initiated by the MacBook program and sent remotely to the VMs. This process is repeated 10 times for each caller–callee configuration. Each call produces two synchronized packet traces collected at the caller- and callee-side VMs. For each of the three applications studied (with Zoom exercised separately at the Free and Business tiers), this yields roughly 112 hours of packet traces in total.

3.2 Relay Server Measurement

With the data collected above, we can identify the relay servers and analyze their impact on call performance. Relay server identification from packet traces proceeds in two steps. 1) Extract relay server IP addresses from the captured traffic: packet traces collected at the Azure VMs contain a mixture of signaling, background application, and RTC media traffic. To precisely identify relay server IPs, we focus on RTP media packets, which dominate call traffic and, under our P2P-blocked setup, are necessarily exchanged with relay servers. Following the DPI approach in [8], we locate RTP packets using header matching and RTP-specific fields, and extract their destination IPs as relay server IPs. 2) Determine the geographic locations of the identified relays: we query *ipinfo.io* [2] for database-based estimates and further validate each reported location using the multi-vantage-latency and traceroute-based methods detailed in our technical report [1]. Only locations consistent across these checks are accepted as valid.

To measure the performance impact of the relay server on a call, we use the RTT computed from RTP packets flowing in both directions. This methodology allows us to extract the most information from a call, given the abundant number of RTP packets per call. For every RTP packet detected by the DPI, we extract its send/arrival timestamps at the caller and callee VMs and sum the caller–relay–callee and callee–relay–caller one-way latencies to obtain the packet's end-to-end RTT.[4] The call-level RTT reported in §4.4 is the mean over all RTP packets in the call; each one-minute call yields more than 10,000 RTP packets, making this mean a stable estimate of the call's typical RTT.

[3]We also disable GPS on both phones throughout the experiments: some RTC apps query the device's GPS for geolocation, which would leak the phone's true location and bypass the VPN-based region emulation.

4 RELAY ANALYSIS

This section characterizes the relay infrastructures we uncovered using the measurement methodology in §3. We first present the global relay maps observed in each application (§4.1). We then analyze how relays are targeted for each call (§4.2) and how applications fall back to alternative relays or transport sessions upon failures (§4.3). Finally, we quantify how relay placement and relay-selection policies jointly impact end-to-end call latency (§4.4).[5]

4.1 Relay Map

Figure 1 plots all relay servers we detected for Zoom (free and business), Discord, and WhatsApp.[6] Each red dot corresponds to a geographic location (nearest city/country) where at least one relay IP was observed. We refer to relays co-located in the same geographic location (e.g., within one city) as one *relay node*. We acknowledge that the relay maps may be incomplete: our caller/callee vantage points may not fully cover all possible relay deployments. Nonetheless, these maps are produced under the same measurement conditions across the three applications (with Zoom shown at both the Free and Business tiers), and thus offer a consistent view of the *relative* relay geolocation density and deployment footprint. The relay counts reported here should therefore be interpreted as *lower bounds* on each application's true relay footprint, reflecting measurements conducted between August 2025 and March 2026; the actual deployments may include additional relay locations not reachable from our 13 vantage points.

Zoom provisions two tiers of relay infrastructure, and business accounts are served by substantially more relay nodes. As shown in Figure 1a and Figure 1b, calls between two Zoom Free accounts consistently traverse only two relay BGP prefixes in the U.S., both announced by Zoom's AS30103: `206.247.0.0/21` (Leesburg/Ashburn, east coast) and `144.195.96.0/21` (San Jose, west coast). In contrast, Zoom Business accounts are served by relays in 12 cities spanning multiple continents: the two U.S. nodes above plus 10 additional cities in Europe, Asia-Pacific, and India. The full list of prefixes and locations is provided in our technical report [1]. This tiered deployment is consistent with Zoom's documentation [34] and suggests differentiated relay capacity and regional coverage across subscription levels. Most Zoom Business relays are operated on Zoom-owned infrastructure (AS30103); the exception is Zoom's India footprint, where Mumbai (`170.114.104.0/21`) and

[4]Wireshark stamps each packet from the host VM's system clock, so the two VMs' clocks may be offset by a constant δ. The offset cancels in our estimate: the $+\delta$ bias in the forward one-way delay offsets the $-\delta$ bias in the reverse one-way delay when the two are summed to form RTT.

[5]We focus on latency as our performance metric because it most directly reflects the impact of relay location on a call. Other metrics such as throughput and jitter are confounded by factors orthogonal to relay placement—e.g., video codec adaptation, load balancing, and network conditions at intermediate hops—and therefore provide a less clean signal for comparing relay infrastructures.

[6]For Zoom, the relays shown here include only those used to establish the *initial* media path. We exclude Zoom's fallback relays because they are predominantly hosted on AWS and require additional failure-triggered measurements to reliably enumerate; we analyze Zoom's fallback behavior in §4.3.

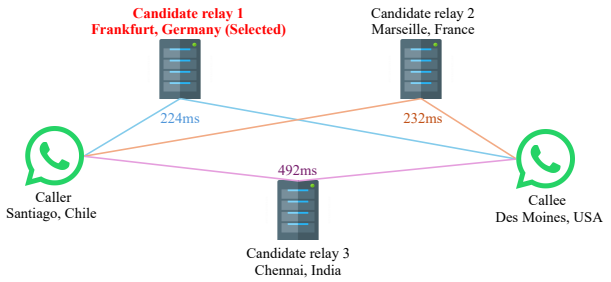

Figure 4: Illustration of WhatsApp's targeting: at call setup, both endpoints send TURN Allocate requests to three candidate relays, and the one that minimizes the combined caller–callee probe RTT is chosen. In this example, the caller (Santiago) and callee (Des Moines) probe three candidates; Frankfurt (224 ms) is picked as the RTT is lower than Marseille (232 ms) and Chennai (492 ms).

Hyderabad (`170.114.112.0/21`) are announced by Oracle Cloud (AS31898), indicating that Zoom leases relay capacity from Oracle to serve Indian users.

Discord operates a globally distributed relay fleet built predominantly on third-party CDN and cloud infrastructure. Discord's relays fall into 31 distinct BGP prefixes spanning 15 cities on every inhabited continent (including Johannesburg); the full list of prefixes and locations is provided in our technical report [1]. The deployment is dominated by Cloudflare PoPs (80% of observations, AS13335), with smaller shares on Google Cloud (16%, AS15169/AS19527) and i3D.net (5%, AS49544, observed only in Tokyo).[7]

WhatsApp exhibits the broadest relay footprint among the studied applications. WhatsApp's relays span 25 cities, with notably higher density in population centers; the full list of IPs, PoP codes, and locations is provided in our technical report [1]. WhatsApp's relays are consistently hosted on Meta-owned infrastructure, indicating a self-operated global relay backbone. The wide geographic spread suggests WhatsApp aims to keep relays close to end users across most regions, which can reduce relay detours and improve call latency—a hypothesis we quantify in §4.4.

4.2 Targeting Strategy

We define *relay targeting* as the policy by which an application selects the geographic relay node that serves a call, given fixed caller and callee locations. Although the global relay map may contain many relay nodes, only a subset (or one) of these nodes are eligible to serve a particular caller–callee pair. We infer targeting behavior by repeating calls under fixed endpoint locations and observing which relay node is ultimately chosen.

Zoom Business and Discord adopt caller-driven targeting. For both, the selected relay is determined solely by the caller's location.[8] Under a fixed caller region, repeated calls select IPs within the same prefix and city, and blocking that prefix prevents call establishment, confirming no alternative node is considered. For Discord, we additionally observe RTP packets exchanged between the caller and relay *before* the callee answers, indicating that relay selection completes entirely on the caller side—a design that may reduce perceived setup latency.

WhatsApp adopts joint caller–callee-aware targeting. As shown in Figure 4, at call setup, both endpoints receive the same set of three TURN server IPs, and each device sends a TURN Allocate request to each candidate and receives the corresponding response. The relay selected for the call is the one minimizing the combined latency observed by the caller and callee, effectively estimating the end-to-end path cost before media transmission. The three candidates and their combined probe RTTs, enumerated across all 169 caller–callee region pairs we tested, are listed in our technical report [1]. Notably, the three TURN server IPs are not fixed for a given caller–callee pair; both the IP addresses and their geographic locations vary across experiments, suggesting an underlying load-balancing mechanism that dynamically adjusts candidate pools while preserving latency-aware selection.

4.3 Fallback Strategy

We define a *fallback session* as a TCP or UDP connection between a client device and a relay IP during a call. RTC systems typically prefer UDP and fall back to TCP under disruption [5, 23]. We count the number of sessions per call (including the default UDP session) to quantify robustness. To trigger fallback, we block relay IP prefixes at the VM during an ongoing call; if the call does not recover within 30 seconds, we consider it terminated. Our measurement results below reveal that Zoom Business and WhatsApp exhibit substantially richer fallback mechanisms than Zoom Free and Discord.

Zoom Free. One UDP and one TCP session to the same relay IP; after blocking both sessions, the call terminates.

Zoom Business. One UDP and one TCP session to the same relay IP; subsequent blocking triggers additional TCP sessions to AWS-hosted servers that are not used under normal conditions, making fallback effectively unbounded.

Discord. One UDP session only; blocking it immediately terminates the call.

WhatsApp. Three UDP sessions to three candidate relays; upon UDP failure, three corresponding TCP sessions are attempted, yielding up to six sessions before termination.

4.4 Call Latency Analysis

Figure 5 shows the average end-to-end RTT for all 13×13 caller–callee region pairs across the three applications. The heatmaps may be asymmetric across the diagonal due to factors such as caller-driven targeting, load-balancing, and asymmetric underlay routing.

A denser relay deployment primarily benefits geographically close calls. We split the 169 region pairs into *intra-continent* (39 pairs whose two endpoints share a *Global Region* entry in Table 1; e.g., `east-us–west-us`, both under *America*) and *inter-continent* (130 pairs whose endpoints fall under different *Global Region* entries). For a fair comparison, we fix each caller–callee pair and measure *per-pair* how much faster Zoom Business, Discord, and WhatsApp are than Zoom Free on the same pair, then average those per-pair deltas within each group. On intra-continent pairs,

[7] Our Discord measurements focus on 1-on-1 calls. Discord channel calls (meeting-room style) use separate media servers with a broader deployment footprint; see Discord's published RTC latency dashboard at `https://latency.discord.media/rtc`.

[8] Zoom Free uses only two relay nodes, making targeting analysis less meaningful; we therefore focus on Zoom Business.

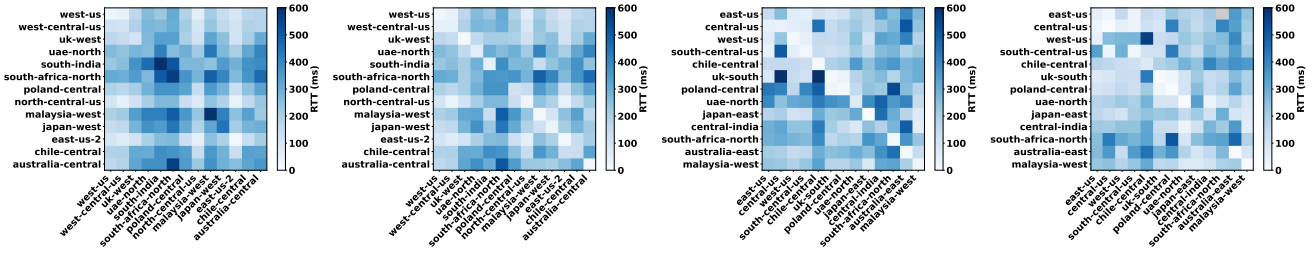

Figure 5: End-to-end call RTT (ms) across 13×13 caller–callee region pairs. Darker cells ■ indicate higher RTT; lighter cells □ indicate lower RTT. Overall, Zoom Free > Zoom Business ≈ Discord > WhatsApp: Zoom Free is the slowest because of its sparse relay footprint (only two U.S. nodes), while WhatsApp is the fastest because its joint caller–callee relay targeting consistently selects a near-optimal relay for each call.

Zoom Business / Discord / WhatsApp are on average **117.2 / 119.7 / 144.0 ms** faster than Zoom Free (**47–58%** per-pair reduction). On inter-continent pairs, however, the savings shrink to **14.7 / 8.0 / 46.7 ms** (**3–17%**): a denser relay map shortens the client–relay leg that dominates short-distance calls, whereas the long-haul underlay sets a hard floor that denser relays cannot overcome for long-distance calls.

A smarter targeting strategy further reduces latency on top of a dense deployment. Zoom Business and Discord have comparably broad global footprints and both use caller-driven targeting; holding the caller–callee pair fixed and averaging the per-pair RTT delta, they differ by only **4.5 ms** on average, despite Zoom Business running on self-owned infrastructure and Discord on Cloudflare. WhatsApp instead adopts joint caller–callee-aware targeting (§4.2), picking the relay minimizing the *combined* caller–callee path cost. For a fair comparison, we again fix each caller–callee pair and measure the per-pair RTT delta: WhatsApp is on average **30.8 ms / 35.4 ms** faster than Zoom Business / Discord per pair. We attribute this additional advantage to WhatsApp balancing both legs rather than conditioning only on the caller's location.

5 CONCLUSION AND DISCUSSION

We measured the relay infrastructures of Zoom, Discord, and WhatsApp, and found that both denser relay deployment and smarter relay targeting are effective levers for reducing end-to-end call latency. The three applications also differ in how relays are hosted—Zoom and WhatsApp operate self-owned relay backbones, while Discord outsources to Cloudflare and other cloud/CDN edges—and in fallback robustness, ranging from WhatsApp's 3-relay UDP+TCP multi-session design to Discord's single-UDP session with no fallback. Our measurements were conducted between August 2025 and March 2026; as providers expand or reconfigure their relay footprints over time, specific numbers reported here (relay counts, city lists, targeting behavior) may shift. Building on this study, we encourage future measurements to further broaden the view along three directions. First, *longitudinal coverage*: repeating measurements over extended time periods would reveal how relay infrastructures and targeting strategies evolve, reflecting providers' infrastructure investments and changes in service policies. Second, *more representative vantage points*: our 13 Azure regions are concentrated in densely populated urban areas; extending measurements to remote or underserved regions would expose relay behavior for users in areas with limited cloud infrastructure, which our current methodology cannot reach. Third, *broader device diversity*: our measurements used iPhone 11 handsets; future work should examine whether relay behavior differs across device types (e.g., Android, laptop clients) or account configurations. Beyond measurement scope, relay infrastructure design may also be shaped by an application's intended use case and user base—e.g., enterprise meeting platforms such as Google Meet or Microsoft Teams may adopt fundamentally different relay strategies than consumer messaging applications; studying these applications is an important direction for future work.

ACKNOWLEDGMENTS

We thank the anonymous ANRW reviewers for their thorough comments and feedback. Liu and Chen were supported in part by NSF grants IIS-2544434 and CNS-2415754. We also thank Jiachen Lu, Patrick Gough, Nengneng Yu, David Dong, and Ankit Penmatcha for their kind help.

REFERENCES

[1] [n. d.]. RTC Relay Server Infrastructure Study: Technical Report. https://github.com/KaiserV2/RTC_relay_server_infra. Extended version with relay IP/prefix tables and the full per-call candidate list..

[2] 2025. IPinfo. https://ipinfo.io/. Accessed: 2025-09-25.

[3] Abdullah Azfar, Kim-Kwang Raymond Choo, and Lin Liu. 2016. Android mobile VoIP apps: a survey and examination of their security and privacy. *Electronic Commerce Research* 16, 1 (mar 2016), 73–111. https://doi.org/10.1007/s10660-015-9208-1

[4] Anastasia Belyh. 2023. Video conferencing statistics for 2024. https://www.founderjar.com/video-conferencing-statistics/

[5] Sumanth Bhat, Ashwin Rao, and Henning Schulzrinne. 2023. Measuring WebRTC Performance in the Wild. In *Proceedings of the ACM Internet Measurement Conference*.

[6] Xiao Sophia Chang, Bernard Jansen, and Zhi-Li Zhang. 2021. Can You See Me Now? A Measurement Study of Zoom, Webex, and Meet. In *Proceedings of the ACM Internet Measurement Conference (IMC)*.

[7] Kai Chen et al. 2018. A Measurement Study of Skype Call Performance. In *Proceedings of the ACM Internet Measurement Conference (IMC)*.

[8] Peiqing Chen, Peng Qiu, Lambda, and Zaoxing Liu. 2025. Protocol Compliance in Popular RTC Applications. In *Proceedings of the 2025 ACM Internet Measurement Conference*. 294–307.

[9] Discord. 2022. Discord Introduction. https://discord.com/company

[10] Jason A. Donenfeld. 2017. WireGuard: Next Generation Kernel Network Tunnel. In *NDSS Workshop on Binary Analysis Research (BAR)*. https://www.wireguard.com/papers/wireguard.pdf

[11] Jordan Holland, Theophilus Benson, and Zhi-Li Zhang. 2017. An Empirical Study of Enterprise Network Traffic. In *Proceedings of the ACM Internet Measurement Conference (IMC)*.

[12] Christer Holmberg, Stefan Hakansson, and Goran Eriksson. 2015. *Web real-time communication use cases and requirements*. Technical Report.

[13] J. Chesterfield J. Ott. 2010. RTP Control Protocol (RTCP) Extensions for Single-Source Multicast Sessions with Unicast Feedback. RFC 5760. https://datatracker.ietf.org/doc/html/rfc5760

[14] Sagar Joshi. 2024. 50 VoIP Statistics to Reveal the Future of Phone Systems. https://learn.g2.com/voip-statistics

[15] Ari Keränen, Christer Holmberg, and Jonathan Rosenberg. 2018. Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal. RFC 8445. https://doi.org/10.17487/RFC8445

[16] Kenji Leibnitz, Tobias Hoßfeld, Naoki Wakamiya, and Masayuki Murata. 2007. Peer-to-peer vs. client/server: Reliability and efficiency of a content distribution service. In *Managing Traffic Performance in Converged Networks: 20th International Teletraffic Congress, ITC20 2007, Ottawa, Canada, June 17-21, 2007. Proceedings*. Springer, 1161–1172.

[17] László Levai et al. 2023. Supercharge WebRTC: Accelerate TURN Services with eBPF/XDP. In *Proceedings of the ACM SIGCOMM Workshop on eBPF and XDP*.

[18] Kyle MacMillan, Salman A. Baset, and Henning Schulzrinne. 2021. Measuring the Performance and Network Utilization of Popular Video Conferencing Applications. In *Proceedings of the ACM Internet Measurement Conference (IMC)*.

[19] Philip Matthews, Jonathan Rosenberg, Dan Wing, and Rohan Mahy. 2008. Session Traversal Utilities for NAT (STUN). RFC 5389. https://doi.org/10.17487/RFC5389

[20] Oliver Michel, Satadal Sengupta, Hyojoon Kim, Ravi Netravali, and Jennifer Rexford. 2022. Enabling passive measurement of zoom performance in production networks. In *Proceedings of the 22nd ACM Internet Measurement Conference* (Nice, France) *(IMC '22)*. Association for Computing Machinery, New York, NY, USA, 244–260. https://doi.org/10.1145/3517745.3561414

[21] Ani Petrosyan. 2022. U.S. video call service usage during COVID-19 2020. https://www.statista.com/statistics/1119981/videoconferencing-services-us-coronavirus-pandemic/

[22] Azeez Raji, Mark Allman, and Robert Beverly. 2021. Measuring the Deployment and Effectiveness of NAT Traversal Techniques. In *Proceedings of the ACM Internet Measurement Conference (IMC)*.

[23] Tirumaleswar Reddy.K, Alan Johnston, Philip Matthews, and Jonathan Rosenberg. 2020. Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN). RFC 8656. https://doi.org/10.17487/RFC8656

[24] Jonathan Rosenberg, Joel Weinberger, Christian Huitema, and Rohan Mahy. 2003. STUN: Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs). In *Proceedings of the Internet Society Symposium on Network and Distributed System Security*.

[25] Tommy Sandholm and Aki Virtanen. 2013. An Analysis of Firewall Traversal Techniques for Real-Time Communication. In *Proceedings of the IEEE International Conference on Communications (ICC)*.

[26] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. 2003. RTP: A Transport Protocol for Real-Time Applications. RFC 3550. https://datatracker.ietf.org/doc/html/rfc3550

[27] Varun Singh and Jörg Ott. 2015. WebRTC: A Platform for Real-Time Communication on the Web. In *Proceedings of the ACM SIGCOMM Computer Communication Review*.

[28] Anirudh Sivaraman et al. 2022. Enabling Passive Measurement of Zoom Performance in Production Networks. In *Proceedings of the ACM Internet Measurement Conference (IMC)*.

[29] Kate Sukhanova. 2023. The Latest Discord Statistics & Trends for 2023. https://techreport.com/statistics/discord-statistics/

[30] Magnus Westerlund and Christer Burman. 2015. *RTCWeb Selective Forwarding Unit (SFU) Architecture*. RFC 7667. IETF.

[31] Zoom. 2026. Plans & Pricing for Zoom Workplace. https://zoom.us/pricing Accessed: 2026-02-09.

[32] Zoom. 2026. Zoom Meetings IP Address Ranges. https://assets.zoom.us/docs/ipranges/ZoomMeetings.txt Accessed: 2026-02-09.

[33] Zoom Support. 2026. Plan Types. https://support.zoom.com/hc/en/article?id=zm_kb&sysparm_article=KB0060423 Accessed: 2026-02-09.

[34] Zoom Support. 2026. Selecting data center for meetings, webinars, whiteboards, notes and docs. https://support.zoom.com/hc/en/article?id=zm_kb&sysparm_article=KB0060026 Accessed: 2026-02-09.