# Session Initiation Protocol (SIP) Technical Overview

By ClearlyIP Published March 11, 2025 40 min read


# Deep Dip in the SIP (Session Initiation Protocol)

## Introduction

Session Initiation Protocol (SIP) is an **application-layer signaling protocol** for establishing, modifying, and terminating real-time communication sessions over IP networks (Source: ietf.org). These sessions can include voice calls, video conferences, multimedia distribution, and other forms of collaborative communication. SIP itself deals only with the control signaling – it sets up and tears

down sessions – while the actual media (audio, video, etc.) is carried by other protocols once a session is established. SIP gained prominence as the de-facto standard for Voice over IP (VoIP) communications, replacing many proprietary protocols and enabling interoperability across different vendors and networks. Over the last two decades, SIP has been widely adopted in telecom infrastructure, enterprise phone systems, and Internet communication services. This report provides a deep technical dive into SIP's current state, including **recent usage trends**, **security vulnerabilities**, **protocol architecture and operations**, and the impact of emerging technologies on SIP's future.

## Recent Trends in SIP Usage and Performance

SIP has been the backbone of IP telephony and session control in unified communications, but its role is evolving amid new technologies and changing usage patterns. **Enterprise telephony and SIP trunking** continue to grow steadily – for example, the global SIP trunking services market grew from USD 7.4 billion in 2018 to an estimated USD 12.7 billion by 2023 (a CAGR of ~11.4%) (Source: didforsale.com). Businesses are increasingly adopting cloud-based SIP trunking solutions to reduce costs and improve scalability (Source: didforsale.com). Integration of SIP with Unified Communications (UC) platforms is a notable trend; SIP trunking is more tightly coupled with UC systems to unify voice, video, and messaging on a single platform, improving collaboration and productivity (Source: didforsale.com)(Source: didforsale.com). There is also growth in leveraging SIP for **video conferencing** capabilities as remote work drives demand for virtual meetings (Source: didforsale.com). In short, in traditional telecom and enterprise domains, SIP usage remains strong and is **considered critical to modern business communications**, providing cost savings, security improvements, and enhanced collaboration features (Source: didforsale.com).

However, SIP's dominance is being challenged in certain areas. The COVID-19 pandemic accelerated the rise of **alternative communication platforms** (Zoom, Teams, WhatsApp, etc.) which often use proprietary or WebRTC-based signaling instead of standard SIP (Source: blog.portaone.com)(Source: blog.portaone.com). Major tech companies (Google, Apple, Facebook, Microsoft, etc.) pushed their own real-time communication solutions, pressuring smaller VoIP vendors to innovate or risk obsolescence (Source: blog.portaone.com). Some industry commentators even provocatively claim *"SIP is dead, long live WebRTC"*, suggesting that WebRTC-based ecosystems will replace traditional SIP telephony in many use cases (Source: blog.portaone.com). Indeed, **WebRTC** (Web Real-Time Communications) has enabled voice/video

directly in browsers and mobile apps, reducing reliance on standalone SIP softphones. A case in point: by 2023, demand for embedded video and web conferencing (often WebRTC-powered) has surged, taking share from conventional SIP-based systems (Source: moldstud.com).

Despite these shifts, it's important to note that **SIP is far from obsolete**. Experts emphasize that while WebRTC and SIP have overlapping use cases, *"one does not clearly replace the other"* (Source: nojitter.com)(Source: nojitter.com). WebRTC lacks a built-in signaling protocol, and often a SIP-based backend or gateway is used to handle call setup even for WebRTC clients (Source: nojitter.com)(Source: nojitter.com). In practice, SIP and WebRTC frequently **complement** each other – WebRTC provides a rich media engine in browsers, and SIP provides robust call signaling and integration with telephony networks (Source: nojitter.com)(Source: nojitter.com). Many "WebRTC calls" still use SIP on the server side (or connect to SIP networks for PSTN access). As a No Jitter industry analysis put it, WebRTC will not *"replace the need for SIP or a SIP-like protocol"*; both technologies have their distinct roles and can **peacefully coexist**, with ample opportunities for synergy (Source: nojitter.com)(Source: nojitter.com).

In terms of **performance**, research indicates there can be differences between pure WebRTC implementations and SIP-based setups. One network comparison study for audio/video conferencing found that under certain test conditions, **WebRTC achieved better throughput, lower jitter, and less packet loss than a SIP-based system**, attributed to factors like codec selection, platform optimizations, and routing strategies (Source: researchgate.net). For instance, WebRTC typically uses the Opus codec and advanced jitter buffer management, which can outperform legacy codecs often used in SIP trunks. This suggests that traditional SIP platforms may need to incorporate newer technologies (e.g. modern codecs, adaptive transports) to match the performance that WebRTC applications deliver. Indeed, the SIP community is responding – there are ongoing efforts to improve SIP's efficiency and adaptability, such as exploring **SIP over QUIC** transport to reduce latency and overhead. An IETF draft (Hurst, 2022) proposes mapping SIP to run over the QUIC protocol (the UDP-based, multiplexed transport behind HTTP/3) for faster, more reliable signaling. Early results indicate that *"SIP-over-QUIC enables a more efficient use of network resources by introducing field compression to the header fields carried in SIP transactions."* (Source: ietf.org) This could significantly cut down SIP's verbose text-based overhead and improve performance in high-volume scenarios. In summary, SIP usage is not in free fall – it remains **widely deployed and growing in absolute terms**, especially for enterprise voice and telecom interconnect – but it is being augmented and reshaped by new technologies to meet modern performance and user experience expectations.

# Security Vulnerabilities and Evolving Attack Vectors

Like all Internet-facing protocols, SIP has its share of security challenges. In fact, the majority of security incidents in VoIP networks today are related to **SIP-based cyberattacks** targeting weaknesses in SIP implementations (Source: audiocodes.com). Attackers often exploit the very openness and flexibility that make SIP powerful – its text-based messages, multiple protocol extensions, and need to interface with numerous devices – to find vulnerabilities. Some of the key SIP security threats and attack vectors include:

- **Denial of Service (DoS) and Distributed DoS (DDoS):** Flooding a SIP server (proxy, registrar, or SBC) with a barrage of SIP messages (e.g. INVITE floods or REGISTER floods) can overwhelm processing and knock voice services offline (Source: audiocodes.com). SIP's transaction-handling mechanisms mean even unsolicited requests elicit some response, so an attacker can cause resource exhaustion. SIP flooding attacks lead to call processing delays, dropped calls, or total service interruption. In severe cases, such attacks have caused multi-hour outages for VoIP service providers.

- **Brute-Force and Authentication Attacks:** Many SIP endpoints (IP phones, PBXs, etc.) use digest authentication for registration and call setup. Attackers frequently attempt password guessing or **credential brute-forcing** against SIP accounts (similar to web login brute force). Unsecured voicemail PINs or weak SIP user passwords can be cracked, leading to unauthorized use of services. **Registration hijacking** is a related threat: an attacker who obtains credentials can register a rogue device as a legitimate user, intercepting incoming calls or making fraudulent outbound calls. Studies of real-world SIP attack traffic have observed frequent brute-force login attempts on SIP servers (Source: mdpi.com)(Source: mdpi.com).

- ** SIP Message Tampering and Injection:** Because SIP messages are text-based, they may be vulnerable to injection attacks or maliciously crafted messages. A classic example is sending **malformed SIP messages** designed to exploit parser bugs (buffer overflows, etc.) in SIP software (Source: mdpi.com). Researchers have demonstrated that fuzzing SIP message fields can crash certain user agents or proxies. Additionally, an attacker could inject commands via SIP headers if an implementation improperly trusts or concatenates header data (though robust parsing and RFC conformance usually mitigate this).

- **Eavesdropping and Man-in-the-Middle (MitM):** If SIP signaling or the associated media streams (RTP) are not encrypted, an attacker with network access can intercept calls or sensitive data. Tools exist to sniff SIP traffic on LANs and reconstruct audio from intercepted RTP packets. **SIP messages can reveal caller/callee identities, DTMF tones (which may**

**convey PINs), or session metadata** if not protected. Man-in-the-middle attackers can potentially alter SIP messages in transit – for example, manipulating the SDP to redirect media streams. The use of Transport Layer Security (TLS) for SIP (SIPS protocol) and Secure RTP (SRTP) for media is increasingly critical to prevent interception and tampering (Source: didforsale.com).

- **Spam and Fraud (SPIT and Toll Fraud):** Unwanted bulk calling (Spam over Internet Telephony, or SPIT) is an emerging nuisance, where attackers use automated SIP dialers to send voice spam (robocalls) or voicemails. More damaging is **toll fraud**, where compromised SIP accounts or PBXs are used to route expensive long-distance calls that attackers profit from. SIP's openness and global reach make it a target for fraudsters who hijack SIP systems to generate calls to premium-rate numbers. Weak authentication or misconfigured outbound dialing rules can leave systems vulnerable to such abuse.

The consequences of successful SIP attacks are serious – they range from **service disruption and downtime** (affecting business continuity) to **call hijacking and impersonation**, **fraudulent call charges**, and **privacy breaches** (Source: mdpi.com)(Source: audiocodes.com). For example, a SIP DoS attack on an emergency call center could block incoming calls, or a hijacked SIP trunk could rack up huge fraudulent bills in minutes. Given these risks, **strengthening SIP security** is a high priority for network operators.

Defenders are employing multiple strategies to combat evolving SIP threats. **Session Border Controllers (SBCs)** and VoIP-aware firewalls are deployed at network perimeters to perform protocol-level filtering, rate limiting, and topology hiding. The SBC terminates and re-originates SIP sessions, masking internal network elements and absorbing abnormal traffic spikes – essentially acting as a security gateway for SIP. SIP servers also implement **intrusion detection rules** (e.g. fail2ban style IP blocking after repeated failed registers) to thwart brute force attempts. Many providers enforce strong authentication (complex SIP passwords or mutual TLS with client certificates). **Encryption** is increasingly standard: SIP over TLS protects signaling, and SRTP (often with secure key exchange via SIP/SDP using DTLS or SDES) encrypts the media, foiling eavesdroppers (Source: didforsale.com). Furthermore, research continues on advanced solutions like anomaly-based attack detection using machine learning (Source: mdpi.com)(Source: mdpi.com), and even blockchain-based authentication schemes for SIP registration (Source: mdpi.com)(Source: mdpi.com).

Despite new attack vectors, it's worth noting that SIP is also evolving with more secure variants and best practices. The SIP specification and related RFCs have incorporated security mechanisms (HTTP digest auth, S/MIME for message bodies, etc.), and newer extensions like **Secure SIP (SIPS**

**URI scheme)** enforce transport-level encryption. As the telecom industry moves toward all-IP networks, the security of SIP signaling is under intense scrutiny, especially for critical services. For example, in 4G/5G mobile networks (IMS-based VoLTE/VoNR), **SIP is used to set up voice calls** and must be protected over the air and in transit – typically IMS signaling is carried over IPsec tunnels or other secure channels in the carrier domain. In summary, SIP's ubiquity makes it a prime target, but the protocol has proven adaptable with layered defenses. Deployments that follow well-established security guidelines – using SBCs, encryption, authentication, and continuous monitoring – can successfully mitigate most SIP-specific threats (Source: audiocodes.com)(Source: audiocodes.com).

## SIP Architecture and Operation in Depth

SIP is a **text-based, client-server protocol** that borrows many concepts from HTTP. A SIP conversation consists of requests sent by a client (for example, an IP phone or softphone initiating a call) and responses from a server (which could be a called device or an intermediary). The core elements of SIP's architecture include **User Agents**, **Proxy Servers**, **Registrars**, and more, each playing specific roles in the signaling process.

**User Agents (UA):** These are the endpoints of communication – typically SIP telephones, softphone apps, or gateways. A user agent has two logical components: a *User Agent Client (UAC)*, which initiates requests, and a *User Agent Server (UAS)*, which responds to requeststelecomtechsource.blogspot.com. In any given call flow, each end acts as UAC for outbound requests and UAS for inbound requests. For example, if Alice calls Bob, Alice's phone acts as a UAC sending an INVITE, and Bob's phone acts as a UAS to accept (or reject) the call. The roles can switch within the dialog: if Bob later sends a BYE to end the call, Bob's UA acts as UAC for that request while Alice's becomes the UAS to respond. This flexible UA model means any SIP endpoint can both initiate and receive sessions.

**Proxy Servers:** A proxy is an intermediary entity that routes SIP requests on behalf of user agents. Per RFC 3261, *"a proxy server is an intermediary that acts as both a server and a client for the purpose of making requests on behalf of other clients"*, mainly to ensure requests are delivered to the next hop toward the target user (Source: onsip.com). In practical terms, a proxy server in a SIP network receives requests (like INVITEs from callers) and determines where to send them – this could be another proxy closer to the callee, or directly to the callee's user agent if known. Proxies handle **routing logic**, enforce policies (e.g. permission to make certain calls), and can modify certain SIP headers as needed (for example, adding record-route headers, or doing NAT address

fix-ups). It's important to note that the distinction between different types of SIP servers is **logical, not physical** – the same software instance could act as a proxy, registrar, and redirect server simultaneously (Source: onsip.com). In many deployments (especially smaller ones), a single server performs multiple roles.

**Registrar Servers:** A registrar is a server that handles **registration** of user locations. When a user agent (say a SIP phone) comes online or moves IP address, it sends a REGISTER request to the registrar, which **associates the user's SIP address (URI) with their current network address** (Source: onsip.com). This information is stored in a **location service** database. For example, Bob's SIP URI might be `sip:bob@example.com`; when Bob's softphone registers from a device, the registrar for *example.com* notes the device's IP and port. Later, when someone tries to call Bob's URI, the proxy for example.com will query the location service to find where Bob is registered and route the INVITE there (Source: docs.oracle.com)(Source: docs.oracle.com). Registrars thus enable user mobility and dynamic mapping of addresses, which is crucial in IP networks where devices' IPs can change. SIP mandates that user agents periodically refresh their registration (with an *Expires* timer) and deregister when going offline, to keep the location service updated. The registration process is secured by authentication to prevent unauthorized bindings.

**Redirect Servers:** A redirect server is a component that does not forward requests like a proxy, but instead **responds to the caller with a new address to try**. It essentially directs the UAC to contact an alternate location. For instance, if a user has moved to a new domain or has multiple registered devices, a redirect server can send a 3xx class response (e.g. "302 Moved Temporarily") pointing to a different URI (Source: onsip.com). The calling UA then re-initiates the request to the provided address. Redirect servers offload proxy load by handling simple rerouting logic, and they allow the endpoints to directly connect on the next attempt. Many SIP services use redirection for least-cost routing or to delegate call setup to another server cluster.

**Back-to-Back User Agents (B2BUA):** Not mentioned in the user's list but worth noting, a B2BUA is a SIP element that acts **as a UAS on one side and UAC on the other**, essentially terminating the call on itself and creating a new call leg to the other party. B2BUAs (like SBCs or PBXs) can provide services like call transfer, topology hiding, and protocol interworking by breaking the end-to-end signaling into two separate SIP dialogs.

These components work together in a typical SIP deployment. **Figure 1** below illustrates a **basic SIP call flow** between two endpoints (UA1 and UA2) via their respective proxy servers, often depicted as the SIP "trapezoid" model (Source: datatracker.ietf.org)(Source: datatracker.ietf.org). The caller (UA1 in domain A) sends an INVITE to Proxy 1 (their outbound proxy). Proxy 1 consults its routing logic and the location service (if the target is in its domain) – in this case, determining that the callee

is in another domain – and forwards the INVITE to Proxy 2 in domain B. Proxy 2 then forwards it to the callee's UA2. Responses (100 Trying, 180 Ringing, etc.) go back along the reverse path. Once the call is established, media will flow directly between UA1 and UA2 (unless proxies are set to stay in the media path, which is uncommon for pure proxies). This trapezoidal routing provides flexibility and scalability: each domain's proxy handles its own users and trusts the other domain's proxy to forward or redirect as needed.

*Figure 1: SIP call flow and trapezoid model — a caller (User Agent A) initiates a session to a callee (User Agent B) through two proxy servers (Proxy A and Proxy B). The ladder diagram shows SIP request/response exchanges: INVITE from A → proxies → B, provisional responses (100 Trying, 180 Ringing) coming back, a 200 OK answer from B, and an ACK from A to confirm session establishment. Media (RTP) flows directly between A and B after the call is set up.*

In the above diagram, the sequence is as follows: **(1)** UA-A sends an `INVITE` to Proxy A (its outbound proxy). **(2)** Proxy A immediately returns a `100 Trying` to UA-A to acknowledge it received the INVITE (Source: cisco.com)(Source: cisco.com). Proxy A then routes the INVITE towards UA-B's domain. **(3)** Proxy A forwards the `INVITE` to Proxy B (the proxy authoritative for UA-B's domain) (Source: cisco.com). **(4)** Proxy B, upon receiving the INVITE, sends its own `100 Trying` back to Proxy A (Source: cisco.com). **(5)** UA-B starts ringing, and sends a `180 Ringing` provisional response to Proxy B (Source: cisco.com), which Proxy B forwards to Proxy A **(6)**, and Proxy A then forwards to UA-A (Source: cisco.com)(Source: cisco.com). The caller hears a ringback tone at this stage. **(7)** When UA-B (the callee) answers, it sends a `200 OK` (with SDP media description) to Proxy B (Source: cisco.com)(Source: cisco.com). This 200 OK travels back through Proxy A to UA-A **(8)**. **(9)** UA-A sends an `ACK` to confirm receipt of the 200 OK, via Proxy A to Proxy B (Source: cisco.com)(Source: cisco.com), which completes the handshake – at this point, the call is established and the two UAs begin exchanging RTP media (audio packets) directly. Finally, when the call ends, one side (say UA-A) sends a `BYE` request which also goes via the proxies and gets answered with a `200 OK` to terminate the session (not fully shown in the figure) (Source: cisco.com) (Source: cisco.com). This example illustrates how SIP signaling traverses possibly multiple servers but ultimately coordinates the endpoints to communicate. Each SIP message in the flow contains various headers (Via, To, From, Call-ID, CSeq, Contact, etc.) to ensure it is routed correctly and associated with the right call dialog.

SIP's message system includes a set of standardized **methods (requests)** and **response codes**. Some of the most common SIP request methods are: **INVITE** (initiate a session), **ACK** (acknowledge final response to INVITE), **BYE** (terminate a session), **CANCEL** (cancel a pending INVITE), **REGISTER** (register location), and **OPTIONS** (query capabilities) (Source: onsip.com). There are

others (e.g. REFER, INFO, PRACK, SUBSCRIBE, NOTIFY, MESSAGE, UPDATE) used for more advanced features, totaling 14 primary methods (Source: onsip.com). SIP responses closely mirror HTTP's style with status codes grouped into classes: 1xx (informational), 2xx (success), 3xx (redirection), 4xx (client error), 5xx (server error), 6xx (global failure) (Source: onsip.com)(Source: onsip.com). For example, `180 Ringing` (a 1xx) indicates the callee is being alerted, `486 Busy Here` is a 4xx indicating the callee is busy, etc. These response codes let SIP handle a variety of call scenarios and failure conditions in a standardized way.

It's important to understand **SIP's layering and relationship with other protocols** in the communication stack. **SIP is an application-layer protocol** that can run over different transport protocols, commonly UDP or TCP on port 5060 for plain SIP, and TCP/TLS on port 5061 for encrypted SIP (SIPS) (Source: ietf.org). Unlike traditional telephony signaling which used dedicated circuits, SIP is designed to be transport-agnostic and even works over SCTP or newer transports. Notably, SIP messages often include embedded information about transports (e.g., via headers indicate if UDP or TCP was used). Figure 2 conceptually shows the protocol stack: at the application layer sits SIP for signaling; the **media sessions** that SIP negotiates are carried by **RTP (Real-time Transport Protocol)** at the transport layer (usually over UDP) (Source: videosdk.live). SIP uses a companion protocol, the **Session Description Protocol (SDP)**, to negotiate and agree on media parameters for the session (Source: nojitter.com)(Source: nojitter.com). SDP is not a transport but rather a message body format – an INVITE or 200 OK will carry an SDP payload listing the supported codecs, media types, and connection info for media streams. For example, SDP might indicate that the caller can send/receive audio on IP X port Y using codec G.711 or Opus (Source: videosdk.live) (Source: videosdk.live). The callee's SDP in the 200 OK will confirm the chosen codec and provide its own receive port. Once this exchange is done, both sides know how to send media to each other via RTP.

Crucially, **SIP signaling and media are separate** – SIP sets up the call, but the voice/video packets flow over RTP (and its control protocol RTCP) directly between the endpoints (unless a media proxy or relay is introduced). This separation is by design for scalability: once a call is established, SIP servers can step out of the path and not be burdened by the high-bandwidth media. As networks evolved, secure variants like **SRTP (Secure RTP)** have been employed to encrypt media, and SIP/SDP has extensions for key exchange to facilitate that (Source: videosdk.live)(Source: videosdk.live). The **SIP protocol stack** thus typically includes: SIP over UDP/TCP for signaling, and RTP/RTCP (over UDP) for media, with optional TLS and SRTP adding security layers (Source: videosdk.live). This modular architecture allows, for instance, SIP to establish a video call that uses RTP for video and perhaps a different channel for data sharing, all negotiated via SDP.

SIP's design also accommodates complex network architectures. In large-scale or carrier environments, SIP is the basis for the **IP Multimedia Subsystem (IMS)** architecture – a core part of 4G/5G networks for delivering voice (VoLTE – Voice over LTE, and VoNR – Voice over New Radio in 5G). IMS defines a series of SIP servers (Proxy-CSCF, Serving-CSCF, Interrogating-CSCF, etc.) which handle registration, call routing, and interconnection between operators. Even in these cases, the fundamental SIP principles remain: user agents (the mobile phones' IMS clients) register via a registrar (the P-CSCF/S-CSCF), and calls are set up with SIP INVITEs that traverse through the IMS core. We will discuss more on SIP's role in 5G in a later section.

In summary, SIP's architecture is **distributed and scalable**. It cleanly separates concerns: **address resolution (via registrars)**, **signaling and policy (via proxies/B2BUAs)**, and **media transport (via RTP)**. All components use standard interfaces (defined by IETF RFCs), which has enabled a vibrant multi-vendor ecosystem of SIP hardware and software. Everything from a small office IP-PBX to a carrier's telephony core uses the same fundamental SIP protocols, differing mainly in deployment scale and auxiliary features. This consistency is a major reason SIP won out over earlier protocols – it is simple yet flexible, able to support basic calling as well as video conferencing, instant messaging (with extensions like SIP SIMPLE), and more. The next sections explore how new technologies and deployment environments are impacting this architecture.

## Rise of WebRTC, QUIC, and Alternative Technologies

As the communication landscape evolves, SIP faces both competition and opportunities from new protocols and frameworks. Two prominent developments are **WebRTC** and **QUIC**, which have begun to influence how real-time communications are implemented, especially in web and cloud environments.

**WebRTC (Web Real-Time Communications):** WebRTC is not a single protocol but a collection of APIs and protocols that enable peer-to-peer audio, video, and data communication directly between browsers or other WebRTC-enabled clients. Introduced in the early 2010s, WebRTC fundamentally changed how developers embed voice/video chat into applications – instead of requiring a separate SIP softphone or plugin, any modern web browser can be a voice/video terminal with just JavaScript. Importantly, WebRTC focuses heavily on the media plane (it handles acquiring microphone/camera, encoding media, NAT traversal with ICE/STUN/TURN, and sending media via RTP/RTCP or newer transports) (Source: nojitter.com)(Source: nojitter.com). What WebRTC deliberately does *not* define is the **signaling protocol** for call setup (Source: nojitter.com)(Source: nojitter.com). This was left open by design: different apps have different needs, and imposing SIP in the browser was seen as

too rigid (and problematic given browser page reloads and state management issues) (Source: nojitter.com)(Source: nojitter.com). As a result, many WebRTC applications use proprietary or simplified signaling – some use JSON over WebSockets, some use HTTP polling, etc. But, notably, **SIP is often used as a signaling layer for WebRTC in enterprise or carrier contexts**. For example, a WebRTC-based softphone in a call center might signal through a WebSocket to a SIP gateway which then connects calls via SIP to the PSTN. WebRTC and SIP thus can work in tandem: WebRTC provides the media engine and browser integration, while SIP provides the robust backend call control. In fact, WebRTC itself reuses key SIP concepts – it uses **SDP offer/answer** for media negotiation, just like SIP (Source: nojitter.com). A WebRTC peer connection generates an SDP offer that can be transported via SIP if desired. Many SIP vendors added WebRTC support by essentially bridging WebRTC streams to SIP networks (often called WebRTC-to-SIP gateways), highlighting the complementary nature. As one analysis succinctly put it, asking whether WebRTC will replace SIP is like asking *"apples or oranges?"* – they serve different purposes, and **both are very important in modern communications** (Source: nojitter.com)(Source: nojitter.com). WebRTC has certainly eroded SIP's use for certain direct app-to-app communications (especially consumer apps and internal team collaborations use WebRTC without SIP), but for anything that requires interoperability, telephony features, or integration with phone networks, SIP remains crucial. The two can and do coexist peacefully (Source: nojitter.com). Looking ahead, WebRTC will likely continue to expand real-time comms into more web and IoT applications, potentially generating even more SIP traffic on the backend as those apps still need to interface with legacy systems or each other.

**QUIC and HTTP/3:** QUIC is a newer transport protocol originally designed by Google and now standardized by the IETF (as the basis for HTTP/3). It operates over UDP and provides multiplexed streams, low latency handshakes, and built-in encryption (TLS 1.3 by default). QUIC's relevance to SIP is twofold: as a transport alternative, and as an inspiration for new signaling paradigms. While classic SIP runs over UDP or TCP, researchers have proposed **running SIP directly over QUIC** to take advantage of its performance benefits. The aforementioned IETF draft "SIP-over-QUIC" describes how to map SIP onto QUIC streams (Source: ietf.org). In this approach, each SIP transaction could be a separate QUIC stream, eliminating head-of-line blocking issues (that can occur with SIP over TCP) and providing better congestion control than UDP. Additionally, QUIC's integrated security would mean SIP messages are encrypted without needing a separate TLS layer. Early analyses suggest significant improvements in network efficiency by compressing SIP's verbose headers and piggybacking on QUIC's stream and multiplexing features (Source: ietf.org). Though experimental, this could modernize SIP for high-volume contexts (imagine thousands of simultaneous calls handled by a single QUIC connection to a server farm). Beyond just SIP-over-QUIC, the principles of QUIC are influencing real-time media too. For example, there's work on **RTP over QUIC** (an IETF draft for tunneling RTP media over QUIC) (Source: datatracker.ietf.org). This

could simplify NAT traversal and improve reliability for media compared to plain UDP. If such transports mature, a future SIP implementation might use QUIC for both signaling and media, making sessions more robust to packet loss and latency.

Another angle is **HTTP-based signaling**. Some have speculated about using web protocols (HTTP/2 or HTTP/3) for session control instead of SIP. While no direct replacement has gained wide traction, certain platforms use REST APIs for call control (for instance, Twilio's API abstracts calls in web terms, though behind the scenes they terminate into SIP trunks). Protocols like **XMPP/Jingle** were alternatives for signaling (used in early Google Talk), but SIP outlasted them in adoption. **Matrix** (an open messaging/VoIP protocol) is a modern competitor which uses HTTP/JSON for federation and can carry VoIP signaling – it's used in applications like Element/Matrix for calls, though again often leveraging WebRTC. To date, none of these have dethroned SIP in its core domains; instead, they serve niche use cases or operate at a higher abstraction level.

In summary, **WebRTC has expanded the reach of real-time communications** beyond the SIP universe, enabling many new applications that don't explicitly use SIP at the client. And **QUIC/HTTP3 hints at a next-generation transport** that could eventually be leveraged by SIP or SIP successors to improve performance. The effect on SIP usage is mixed: some communications that would have been SIP-based in the past (say, a quick video chat between two users) might now happen via WebRTC without touching SIP. On the other hand, the overall volume of real-time communications has vastly increased (thanks to WebRTC), and much of it eventually interconnects with SIP networks (for example, a web app calling a phone). Rather than a strict replacement, these technologies are **complementary**. WebRTC handles the last-mile to the user in many modern apps, while SIP continues to do the heavy lifting in the core, stitching together calls between disparate systems. As telecom networks adopt QUIC, we may see SIP itself evolve (or a new protocol emerge) to utilize QUIC's capabilities, ensuring that SIP-based services remain competitive in setup latency and reliability compared to web-native approaches.

## SIP Deployment Challenges in Modern Networks

Implementing SIP at scale – whether in large enterprise systems or cloud-based services – comes with a unique set of challenges. While the protocol is standardized, real-world deployment involves dealing with network quirks, ensuring quality at scale, and integrating with cloud infrastructure. Some of the major **deployment challenges** and considerations include:

- **NAT Traversal and Firewall Issues:** Perhaps the most notorious challenge for SIP is handling Network Address Translation (NAT). SIP was designed assuming end-to-end connectivity, but in practice most users sit behind NAT routers (in homes, offices, cloud VNets, etc.). NATs break SIP in multiple ways: the IP/port in the SIP headers (Contact, Via, SDP's media connection info) may be the *internal* addresses which are unreachable externally. Thus, a callee might try to send RTP to a private IP. Additionally, many NATs are stateful and will block incoming packets unless outgoing traffic has opened a pinhole. **To solve this, several techniques are used:** STUN (Session Traversal Utilities for NAT) allows a SIP UA to discover its public IP/port and include that in SDP; TURN (Traversal Using Relays) provides a media relay server in the cloud to carry RTP when direct P2P path fails; and SIP-aware ALG (Application Layer Gateway) in routers attempt to automatically rewrite SIP packets to fix IPs (though ALGs are often problematic). Proxies and SBCs also assist by rewriting Contact and SDP on the fly – for example, an SBC on the network edge will adjust the SDP to insert its own IP as the media endpoint, thus bridging between internal and external addresses. Despite these, NAT remains the "enemy" – misconfigured NAT or firewall rules are a top cause of one-way audio or failed calls in VoIP deployments (Source: docs.netscaler.com). Modern cloud-native apps often use ICE (Interactive Connectivity Establishment) which systematically tries STUN and TURN candidates to get media through; fortunately, ICE can be adopted in SIP as well (there is an ICE extension for SIP/SDP). Still, administrators must carefully design their network: e.g., ensure SIP signaling ports are open, RTP port ranges are consistent and permitted, and that SIP endpoints know their public address or use a relay.

- **Scalability and Performance:** As the number of users or calls grows, SIP servers must scale to handle registration loads, busy-hour call attempts, and media relay (if any). Unlike web requests, SIP sessions are long-lived (a call might last minutes or hours), which means maintaining state. Scaling **registrar servers** is challenging because the location database must be consistent – solutions include using a distributed database or partitioning by user groups. **Proxy servers** can be stateless or stateful; a stateless proxy forwards messages and forgets, making it easier to scale (just replicating proxies behind a load balancer), but most proxies operate statefully at least per transaction to handle retransmissions and so on. High-performance SIP proxies like Kamailio or OpenSIPS can handle thousands of calls per second on modest hardware, but careful tuning (transaction timers, memory pools) is needed. When scaling out, **load balancing** SIP traffic is tricky: a simple round-robin LB might break SIP's assumption that requests of a dialog go to the same server. Specialized SIP load balancers or hashes based on Call-ID can maintain call stickiness. In cloud deployments, one might use a combination of global load balancers (for redundancy across regions) and redirect servers (to send clients to the optimal proxy). **Throughput and latency** considerations are also vital – each

call setup is a flurry of small messages; proxies must minimize added latency. Co-locating SIP servers strategically (e.g., regional SBCs close to users) helps reduce call setup times (important for user experience – people expect the phone to ring within a second or two of dialing).

- **Reliability and Redundancy:** Telephony systems demand high availability. SIP provides some built-in redundancy mechanisms: clients can use DNS SRV records to discover multiple SIP server IPs for a domain (if one proxy is down, try the next priority) (Source: datatracker.ietf.org) (Source: datatracker.ietf.org). Many deployments use **server clustering** with failover – if a proxy fails mid-call, sometimes dialogs can be recovered on a backup if they share state via a database (although mid-call failover is complex; often calls in progress drop, but new ones go through backup). Registrations and subscriptions need to be replicated to avoid single points of failure. Cloud deployment makes it easier to spin up new instances, but ensuring SIP transactions seamlessly migrate or that clients retry properly is an operational challenge. Protocol extensions like Outbound (RFC 5626) allow a UA to maintain flows to multiple proxies for redundancy. Testing failure scenarios (network splits, etc.) is key: e.g., will phones re-register to a secondary if primary is down? Also, **stateful devices like SBCs** must often come in pairs (active/standby) so that media and signaling can continue on standby when active fails, using techniques like virtual IPs or state synchronization.

- **Interoperability and Standards Compliance:** SIP has many extensions and optional features. Different vendors' equipment sometimes don't play perfectly together – one might encounter issues with how SDP is handled, or proprietary headers. Ensuring interoperability requires adhering strictly to RFC standards and sometimes implementing workarounds (for instance, normalizing SIP header formats, supporting both early media approaches, etc.). Testing against known platforms (Cisco, Avaya, Microsoft Teams Direct Routing, etc.) is often part of deployment to catch interop issues early. For VoIP service providers, Session Border Controllers often perform *SIP normalization*, converting one flavor of SIP to another to connect disparate systems.

- **Quality of Service (QoS) and Network Constraints:** Delivering high audio quality over IP means dealing with jitter, packet loss, and latency. While this is more about RTP (media) than SIP, the signaling has a role in negotiating things like codec (higher compression vs higher quality). In modern networks, ensuring low latency for SIP messages (for fast call setup) and prioritizing RTP packets (so voice has lower jitter) is done via QoS policies. In enterprise networks, VoIP VLANs and DSCP markings (e.g., EF for voice) are used. In cloud or internet scenarios, one has less control, but techniques like forward error correction or adaptive jitter buffers in endpoints alleviate issues. SIP itself can be used with voice quality monitoring: e.g.,

sending SIP INFO or reports with quality metrics, or using RTCP XR (extended reports). A challenge is that as calls traverse multiple networks (Wi-Fi, cellular, internet backbone), ensuring end-to-end QoS is non-trivial – SIP deployment engineers must design for resiliency (e.g., allow codec fallback from HD voice to narrowband if needed, use shorter RTP packets to cope with loss, etc.).

- **Cloud-Native Considerations:** Deploying SIP infrastructure in the cloud (AWS, Azure, GCP or containerized environments) introduces some new wrinkles. Cloud instances often have ephemeral IPs – one must use solutions like elastic IPs or proper DNS so SIP endpoints can register to a stable address and not be affected by autoscaling events. Container orchestration (Kubernetes) can complicate SIP because of dynamic service discovery; however, projects exist to enable SIP in Kubernetes (using StatefulSets for proxy clusters, for example). Another concern is media path in the cloud: hairpinning media into a cloud data center could add latency; some architectures keep media local (e.g., using distributed media relays). Additionally, cloud security mechanisms (like AWS Security Groups) must be carefully opened for the right ports; one misconfiguration and SIP traffic may be unintentionally blocked. On the plus side, cloud deployments benefit from on-demand scalability and global reach – one can deploy SIP servers in multiple regions to serve users closer and achieve resilience.

- **Monitoring and Management:** Operating a SIP network at scale requires robust monitoring of both signaling and media. Tools like SIP trace logs (using SIP capture tools or protocols like HEP), media quality monitoring (MOS scores via RTCP), and system metrics (CPU, memory of proxy servers) are essential. Unlike web services, a SIP call involves multiple network hops and two different paths (signaling vs media), so troubleshooting can be complex. Automation and orchestration are becoming vital – using DevOps practices to manage configurations, deploying updates without downtime, etc., all while ensuring registrations and calls are not disrupted.

In short, SIP can certainly scale to carrier-grade deployments (telecoms handle millions of subscribers with IMS, and major ITSPs serve huge call volumes with SIP), but it requires careful **network architecture and tuning**. Many challenges revolve around the fundamental tension of SIP being an overlay application protocol on an IP network not originally designed for real-time sessions. Through experience and standards evolution, the community has developed solutions (like STUN/TURN for NAT, DNS SRV for redundancy, and so on) to address these challenges. As SIP migrates into fully cloud-based environments, some of these issues (like NAT) might be mitigated by cloud networking improvements, whereas others (scale, automation) will become easier with cloud tooling. The key is that a successful SIP deployment in 2025 must blend **telecom reliability** with **cloud agility**, ensuring that users get a dial-tone service that is as dependable as a traditional phone, but as flexible and scalable as a web service.

# SIP's Future in 5G, Unified Communications, and Telecom Convergence

SIP is a mature protocol, but its journey isn't over – it continues to be a linchpin in new telecommunications advancements. In **5G networks**, which are IP-based end-to-end, SIP retains a central role for providing voice and video calling services. The 5G core network does not define a new call signaling protocol; instead, it leverages the existing **IMS (IP Multimedia Subsystem)** infrastructure from 4G LTE. Voice over 5G (VoNR – Voice over New Radio) is essentially an evolution of VoLTE and relies on the same IMS SIP signaling for call setup and teardowntechplayon.comtechplayon.com. In a VoNR scenario, when a 5G smartphone places a call, the device's IMS client uses SIP to invite the other party, and the call is established over the 5G data network with QoS guarantees. The difference in 5G is mostly in the radio and core support (enabling ultra-low latency and better quality), but from an application perspective, **SIP is still the session control protocol** connecting the call through the IMS core. As Techplayon notes, *"SIP is used for signaling procedures between the UE and IMS"* in VoNR, with dedicated QoS flows for SIP messages to ensure minimal latency for call signalingtechplayon.com. Telecom operators have heavily invested in IMS, and SIP is deeply embedded there – not just for person-to-person calls, but also for services like **Rich Communication Services (RCS)** messaging, video over LTE, and more, which use SIP messaging or its extensions under the hood.

Furthermore, SIP serves as the lingua franca for **interconnecting different service providers' networks**. In both 4G and 5G, when a voice call needs to traverse between carriers (or between an operator and an enterprise VoIP network), it is typically handed over via a SIP trunk or peering interface. For example, Ericsson notes that *"SIP-based interconnect enables VoLTE to interoperate across the boundaries of CSPs (Carriers) using HD voice and video calling"* (Source: ericsson.com). This means two carriers with VoLTE can directly exchange calls as SIP messages over IP, preserving advanced features like wideband voice. The number of networks using such **SIP interconnects is increasing**, and over time, it's expected that most voice traffic globally will be IP-to-IP (SIP) between carriers, rather than falling back to older TDM interconnect (Source: ericsson.com). This trend solidifies SIP's relevance: even as we phase out legacy circuits (PSTN/SS7), SIP is the protocol carrying voice through the internet and between telecoms.

In the **enterprise unified communications** space, SIP also remains foundational. Unified Communications (UC) platforms – think of systems like Cisco Unified Communications Manager, Microsoft Teams (with Direct Routing), Zoom Phone, Avaya Aura, etc. – all use SIP in one form or another to handle call signaling. Enterprises might not expose SIP to end-users (they click a name in

Teams and a call happens, seemingly without dialing), but behind the scenes **SIP trunking connects these UC systems to the PSTN and to each other**. Microsoft's Direct Routing for Teams, for instance, is essentially a SIP trunk from Teams' cloud to an SBC on premises or in Azure. Cisco phones and telepresence units speak SIP (or the closely related SCCP in some cases, but SIP is common for interoperability). The fact that SIP is an open standard has allowed a huge ecosystem of IP phones, conference devices, and software clients to interoperate – a Polycom phone can register to an Asterisk or to a cloud PBX using SIP, for example. **SIP unified communications features** like presence, instant messaging, and conferencing have been defined (often via extensions or companion protocols like SIMPLE for presence). Admittedly, some UC vendors have proprietary elements, but SIP is typically the baseline for call control and federation. For instance, if an enterprise wants to connect their phone system to a voice gateway or to a SIP-based contact center, SIP is the common language.

One evolving area is **telecom-media convergence and IoT**. As 5G aims to also connect IoT devices and merge various services (voice, data, content delivery), there's been discussion on whether SIP could extend to, say, control certain IoT communications or if lighter protocols will take that role. Largely, voice/video remain SIP's domain, while IoT device signaling might use other protocols (MQTT, etc.). But in scenarios like emergency services, SIP is part of newer Next-Generation 9-1-1 systems to carry not just calls but also real-time text and video to emergency centers.

Another question for SIP's future is whether something will replace it for voice signaling. Some have suggested using pure HTTP/2 or REST for call control, particularly in web-centric environments. While no direct successor has gained momentum yet, one could envision a future "HTTP-based SIP" for simpler integration with web stacks. Until then, the industry momentum behind SIP and its installed base make it likely to persist. The introduction of IMS in 3GPP and heavy reliance on SIP for 5G virtually guarantees that SIP will be in use for at least the lifespan of those technologies (which is easily another decade or more). In addition, the rise of **Network Function Virtualization (NFV)** and cloud-native telecom functions has meant reimplementing SIP servers as virtualized or containerized software, but not eliminating them. Telecom vendors offer virtual SBCs, cloud-native IMS cores, etc., all still speaking SIP but running on modern infrastructure.

In terms of features, SIP is being adapted to new needs: support for richer media (like VR/AR streams) could be negotiated via extensions to SDP, integration with identity frameworks for caller identification (to combat call spoofing, standards like STIR/SHAKEN use SIP headers to carry identity info), and better support for multiparty and collaboration (though that's often handled at application level beyond SIP).

In **summary**, SIP's future looks secure in the realms of **5G voice and beyond**, **interoperable communications**, and as a backbone for enterprise telephony. It provides the glue between traditional telephony and the IP world. While from an end-user perspective the technology may become invisible (users just "click to connect" in an app), SIP will likely be orchestrating the session behind the scenes. As networks converge and everything becomes packet-based, SIP stands as a proven, adaptable protocol to converge voice, video, and messaging across disparate systems. It's telling that even after 20+ years, no replacement protocol has achieved the universality of SIP for session control – instead, we see SIP augmented with modern transports and surrounded by complementary web technologies. Thus, rather than being displaced, SIP is evolving quietly: it may run over QUIC instead of TCP, it may be delivered through serverless cloud functions instead of hardware PBXs, but it will still be *Session Initiation Protocol*, doing what it does best – initiating sessions – for years to come (Source: mdpi.com)(Source: mdpi.com).

## References (Key Sources)

1. J. Rosenberg et al., *RFC 3261: SIP: Session Initiation Protocol*. IETF (2002). – **Definition and core architecture of SIP** (Source: ietf.org)(Source: ietf.org).

2. DidforSale Blog – *2023 SIP Trunking Trends and Predictions*. (Feb 2023). – **Market growth of SIP trunking and future trends** (Source: didforsale.com)(Source: didforsale.com).

3. PortaOne Blog – *SIP Is Dead. Long Live WebRTC!* (2020). – **Perspective on WebRTC vs SIP** (Source: blog.portaone.com).

4. No Jitter – *Will WebRTC Replace SIP?* (2019). – **Analysis of WebRTC and SIP complementarity** (Source: nojitter.com)(Source: nojitter.com).

5. Pereira, D., Oliveira, R. – *Detection of Abnormal SIP Signaling Patterns: A Deep Learning Comparison*. Computers (MDPI) vol.11, no.2 (2022). – **SIP attacks and usage in 4G/5G networks** (Source: mdpi.com)(Source: mdpi.com).

6. AudioCodes Blog – *Securing Your Voice Network: Why SBCs are Essential* (2023). – **VoIP/SIP security threats overview** (Source: audiocodes.com).

7. OnSIP VoIP Resources – *SIP Proxy Server (RFC3261 definitions)*. – **Definitions of SIP Proxy, Registrar, Redirect** (Source: onsip.com)(Source: onsip.com).

8. Oracle Java ME SDK – *Understanding the SIP Registrar and Proxy*. – **Explanation of registrar and proxy roles** (Source: docs.oracle.com)(Source: docs.oracle.com).

9. OnSIP – *SIP Call Flow Explained*. – **Illustration of basic SIP call flow and messages** (Source: onsip.com)(Source: onsip.com).

10. Cisco – *SIP Call Flows (ATA Administration Guide)*. – **Ladder diagram steps of SIP call setup and teardown** (Source: cisco.com)(Source: cisco.com).

11. VideoSDK – *Mastering SIP: Comprehensive Guide*. – **SIP protocol stack and media (RTP/SRTP) separation** (Source: videosdk.live)(Source: videosdk.live).

12. Techplayon – *VoNR Call Flow – Voice over 5G NR*. (2021). – **Confirmation that 5G VoNR uses IMS SIP signaling**techplayon.com.

13. Ericsson White Paper – *Voice and Communication Services in 4G and 5G*. (2020). – **SIP-based interconnect for VoLTE/VoNR and increasing SIP use between carriers** (Source: ericsson.com)(Source: ericsson.com).

14. Research study (Husni et al. 2021) – *Network Performance Comparison of WebRTC and SIP*. – **Finding that WebRTC had better jitter/packet loss performance than SIP in tests** (Source: researchgate.net).

15. IETF Draft – S. Hurst, *SIP-over-QUIC: SIP over QUIC Transport*. (Oct 2022). – **Proposal to run SIP over QUIC for efficiency** (Source: ietf.org).

---

Tags: sip, session initiation protocol, voip, networking, telecom, unified communications, protocol architecture, ip telephony, signaling, security

---

# About ClearlyIP

### ClearlyIP Inc. — Company Profile (June 2025)

---

#### 1. Who they are

ClearlyIP is a privately-held unified-communications (UC) vendor headquartered in Appleton, Wisconsin, with additional offices in Canada and a globally distributed workforce. Founded in 2019 by veteran

FreePBX/Asterisk contributors, the firm follows a "build-and-buy" growth strategy, combining in-house R&D with targeted acquisitions (e.g., the 2023 purchase of Voneto's EPlatform UCaaS). Its mission is to "design and develop the world's most respected VoIP brand" by delivering secure, modern, cloud-first communications that reduce cost and boost collaboration, while its vision focuses on unlocking the full potential of open-source VoIP for organisations of every size. The leadership team collectively brings more than 300 years of telecom experience.

---

## 2. Product portfolio

- **Cloud Solutions** – Including *Clearly Cloud* (flagship UCaaS), **SIP Trunking**, **SendFax.to** cloud fax, **ClusterPBX OEM**, **Business Connect** managed cloud PBX, and **EPlatform** multitenant UCaaS. These provide fully hosted voice, video, chat and collaboration with 100+ features, per-seat licensing, geo-redundant PoPs, built-in call-recording and mobile/desktop apps.
- **On-Site Phone Systems** – Including CIP PBX appliances (FreePBX pre-installed), ClusterPBX Enterprise, and Business Connect (on-prem variant). These offer local survivability for compliance-sensitive sites; appliances start at 25 extensions and scale into HA clusters.
- **IP Phones & Softphones** – Including CIP SIP Desk-phone Series (CIP-25x/27x/28x), fully white-label branding kit, and *Clearly Anywhere* softphone (iOS, Android, desktop). Features zero-touch provisioning via Cloud Device Manager or FreePBX "Clearly Devices" module; Opus, HD-voice, BLF-rich colour LCDs.
- **VoIP Gateways** – Including Analog FXS/FXO models, VoIP Fail-Over Gateway, POTS Replacement (for copper sun-set), and 2-port T1/E1 digital gateway. These bridge legacy endpoints or PSTN circuits to SIP; fail-over models keep 911 active during WAN outages.
- **Emergency Alert Systems** – Including **CodeX** room-status dashboard, **Panic Button**, and **Silent Intercom**. This K-12-focused mass-notification suite integrates with CIP PBX or third-party FreePBX for Alyssa's-Law compliance.
- **Hospitality** – Including **ComXchange** PBX plus PMS integrations, hardware & software assurance plans. Replaces aging Mitel/NEC hotel PBXs; supports guest-room phones, 911 localisation, check-in/out APIs.
- **Device & System Management** – Including **Cloud Device Manager** and **Update Control (Mirror)**. Provides multi-vendor auto-provisioning, firmware management, and secure FreePBX mirror updates.
- **XCast Suite** – Including Hosted PBX, SIP trunking, carrier/call-centre solutions, SOHO plans, and XCL mobile app. Delivers value-oriented, high-volume VoIP from ClearlyIP's carrier network.

---

## 3. Services

- **Telecom Consulting & Custom Development** – FreePBX/Asterisk architecture reviews, mergers & acquisitions diligence, bespoke application builds and Tier-3 support.
- **Regulatory Compliance** – E911 planning plus **Kari's Law**, **Ray Baum's Act** and **Alyssa's Law** solutions; automated dispatchable location tagging.
- **STIR/SHAKEN Certificate Management** – Signing services for Originating Service Providers, helping customers combat robocalling and maintain full attestation.
- **Attestation Lookup Tool** – Free web utility to identify a telephone number's service-provider code and SHAKEN attestation rating.
- **FreePBX® Training** – Three-day administrator boot camps (remote or on-site) covering installation, security hardening and troubleshooting.
- **Partner & OEM Programs** – Wholesale SIP trunk bundles, white-label device programs, and ClusterPBX OEM licensing.

---

**4. Executive management (June 2025)**

- **CEO & Co-Founder: Tony Lewis** – Former CEO of Schmooze Com (FreePBX sponsor); drives vision, acquisitions and channel network.
- **CFO & Co-Founder: Luke Duquaine** – Ex-Sangoma software engineer; oversees finance, international operations and supply-chain.
- **CTO & Co-Founder: Bryan Walters** – Long-time Asterisk contributor; leads product security and cloud architecture.
- **Chief Revenue Officer: Preston McNair** – 25+ years in channel development at Sangoma & Hargray; owns sales, marketing and partner success.
- **Chief Hospitality Strategist: Doug Schwartz** – Former 360 Networks CEO; guides hotel vertical strategy and PMS integrations.
- **Chief Business Development Officer: Bob Webb** – 30+ years telco experience (Nsight/Cellcom); cultivates ILEC/CLEC alliances for Clearly Cloud.
- **Chief Product Officer: Corey McFadden** – Founder of Voneto; architect of EPlatform UCaaS, now shapes ClearlyIP product roadmap.
- **VP Support Services: Lorne Gaetz** (appointed Jul 2024) – Former Sangoma FreePBX lead; builds 24×7 global support organisation.
- **VP Channel Sales: Tracy Liu** (appointed Jun 2024) – Channel-program veteran; expands MSP/VAR ecosystem worldwide.

---

**5. Differentiators**

- **Open-Source DNA:** Deep roots in the FreePBX/Asterisk community allow rapid feature releases and robust interoperability.
- **White-Label Flexibility:** Brandable phones and ClusterPBX OEM let carriers and MSPs present a fully bespoke UCaaS stack.
- **End-to-End Stack:** From hardware endpoints to cloud, gateways and compliance services, ClearlyIP owns every layer, simplifying procurement and support.
- **Education & Safety Focus:** Panic Button, CodeX and e911 tool-sets position the firm strongly in K-12 and public-sector markets.

---

**In summary**

ClearlyIP delivers a comprehensive, modular UC ecosystem—cloud, on-prem and hybrid—backed by a management team with decades of open-source telephony pedigree. Its blend of carrier-grade infrastructure, white-label flexibility and vertical-specific solutions (hospitality, education, emergency-compliance) makes it a compelling option for ITSPs, MSPs and multi-site enterprises seeking modern, secure and cost-effective communications.

---