AWS Whitepaper

Real-Time Communication on AWS

Real-Time Communication on AWS: AWS Whitepaper

Table of Contents

This whitepaper is for historical reference only. Some content might be outdated and some links might not be available.

Real-Time Communication on AWS

Best Practices for Designing Highly Available and Scalable Real-Time Communication (RTC) Workloads on AWS

Publication date: **May 5, 2022** (*Document revisions*)

Abstract

Today, many organizations are looking to reduce cost and attain scalability for real-time voice, messaging, and multimedia workloads. This paper outlines the best practices for managing real-time communication (RTC) workloads on Amazon Web Services (AWS), and includes reference architectures to meet these requirements. This paper serves as a guide for individuals familiar with real-time communication on how to achieve high availability and scalability for these workloads.

This paper includes reference architectures that show how to set up RTC workloads on AWS, and best practices to optimize the solutions to meet end user requirements while optimizing for the cloud. The Evolved Packet Core (EPC) is out of scope for this whitepaper, but the best practices detailed here can be applied to Virtual Network Functions (VNFs).

Are you Well-Architected?

The AWS Well-Architected Framework helps you understand the pros and cons of the decisions you make when building systems in the cloud. The six pillars of the Framework allow you to learn architectural best practices for designing and operating reliable, secure, efficient, cost-effective, and sustainable systems. Using the AWS Well-Architected Tool, available at no charge in the AWS Management Console (sign-in required), you can review your workloads against these best practices by answering a set of questions for each pillar.

For more expert guidance and best practices for your cloud architecture—reference architecture deployments, diagrams, and whitepapers—refer to the AWS Architecture Center.

Introduction

Telecommunication applications using voice, video, and messaging as channels are a key requirement for many organizations and their end users. These real-time communication (RTC) workloads have specific latency and availability requirements that can be met by following relevant design best practices. In the past, RTC workloads have been deployed in traditional on-premises data centers with dedicated resources.

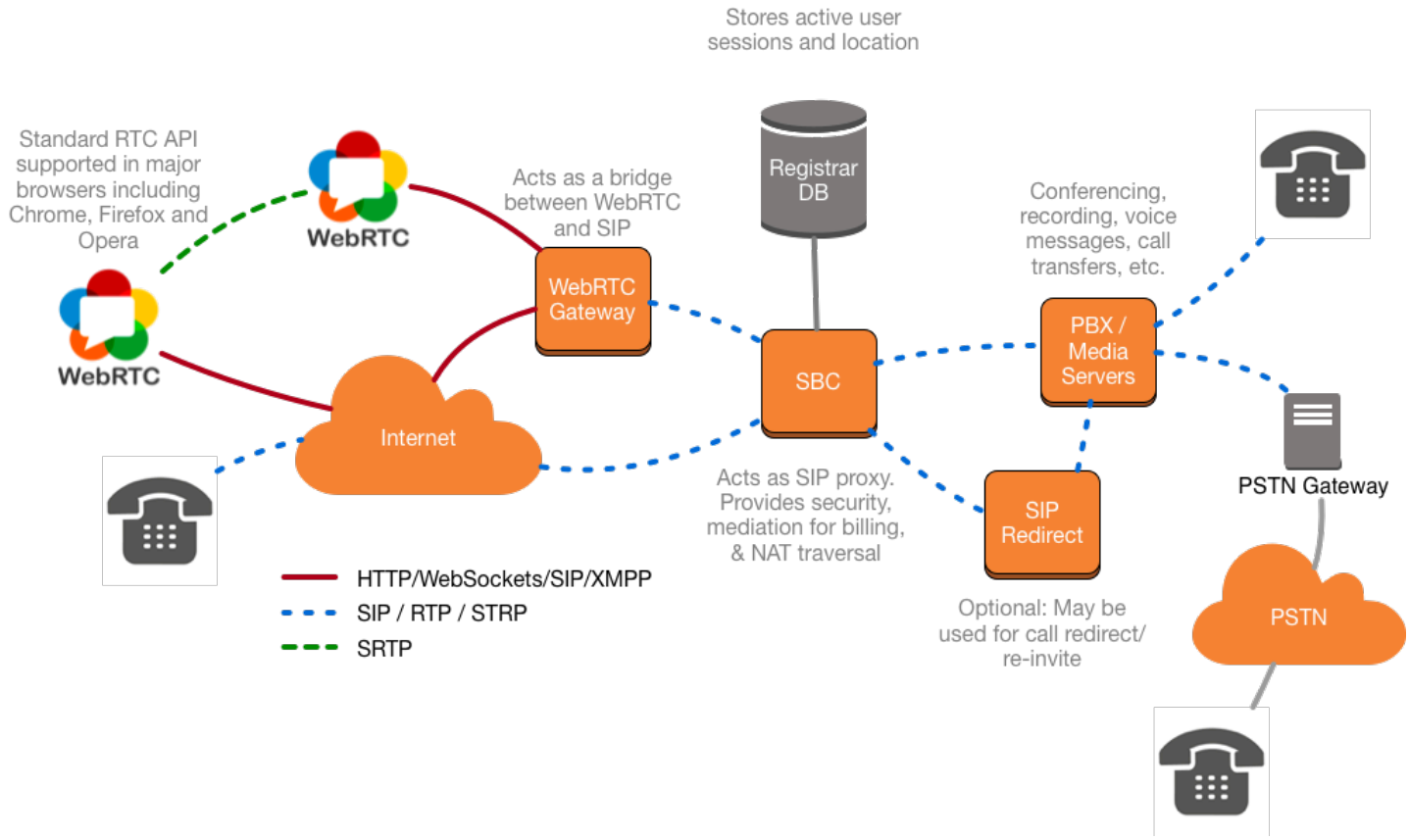
RTC workloads require a highly scalable, resilient, and available environment. Today, customers use AWS to run RTC workloads with reduced cost, improved agility, elasticity, and time to market.

Fundamental components of RTC architecture

In the telecommunications industry, RTC commonly refers to live media sessions between two endpoints with minimum latency. These sessions could be related to:

- A voice session between two parties (such as a telephone system, mobile, or Voice over IP (VoIP))
- Instant messaging (such as chatting and Instant Relay Chat (IRC))
- Live video session (such as video conferencing and telepresence)

Each of the preceding solutions has some components in common (such as components that provide authentication, authorization and access control, transcoding, buffering and relay, and so on) and some components unique to the type of media transmitted (such as broadcast service, messaging server and queues, and so on). This section focuses on defining a voice- and video-based RTC system and all of the related components, as illustrated in the following figure.



Essential architectural components for RTC

Softswitch/PBX

A softswitch or PBX is the brain of a voice telephone system and provides intelligence for establishing, maintaining, and routing of a voice call within or outside the enterprise by using different components. All of the subscribers of the enterprise are required to register with the softswitch to receive or make a call. An important functionality of the softswitch is to keep track of each subscriber and how to reach them by using the other components within the voice network.

Session border controller (SBC)

A session border controller (SBC) sits at the edge of a voice network and keeps track of all incoming and outgoing traffic (both control and data planes). One of the key responsibilities of an SBC is to protect the voice system from malicious use. The SBC can be used to interconnect with session initiation protocol (SIP) trunks for external connectivity. Some SBCs also provide transcoding capabilities for converting CODECs from one format to another. Most SBCs also provide network address translation (NAT) traversal capabilities, which aids in ensuring calls are established, even across firewalled networks.

PSTN connectivity

Voice over IP (VoIP) solutions use Public Switched Telephone Network (PSTN) gateways and SIP trunks to connect with legacy PSTN networks.

PSTN gateway

The PSTN gateway converts the signaling between SIP and SS7 and media between Real Time Transport Protocol (RTP) and time division multiplexing (TDM) using CODEC transcoding. PSTN gateways always sit at the edge close to the PSTN network.

SIP trunk

In a SIP trunk, the enterprise does not end its calls onto a TDM (SS7 based) network, but rather the flows between enterprise and telco remain over IP. Most of the SIP Trunks are established by using SBCs. The enterprise must agree on the predefined security rules from telco, such as allowing a certain range of IP addresses, ports, and so on.

Media gateway (transcoder)

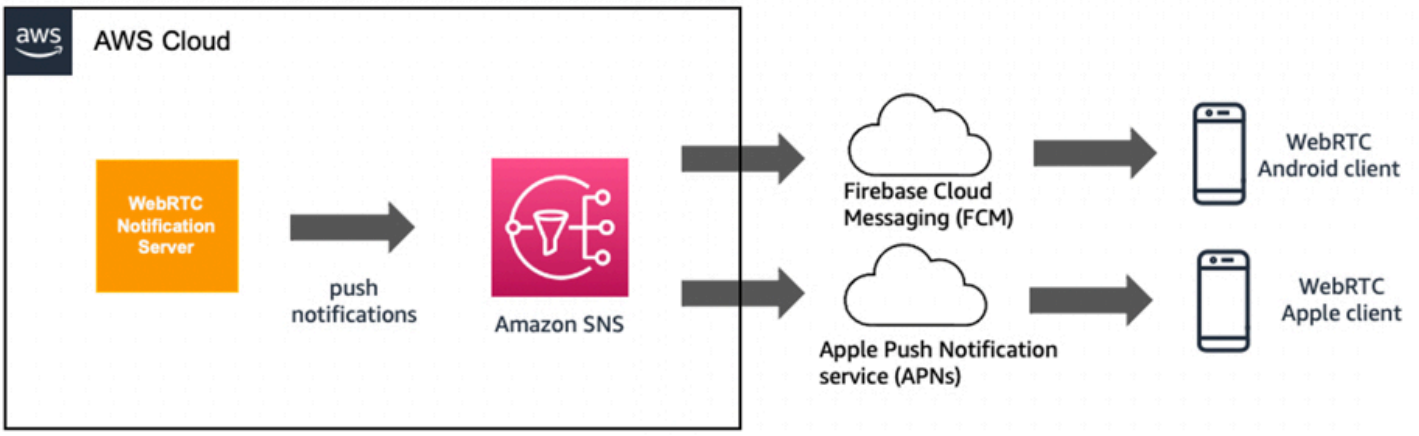
Users communicate in real-time using audio and/or video, as well as optional data and other information. To communicate, the two devices need to be able to agree upon a mutually-understood codec for each media track, so they can successfully communicate and present the shared media. All WebRTC-compatible browsers must support online positioning user support (OPUS) and G711 for audio, VP8, and H.264 Constrained Baseline profile for video.

A typical voice solution outside the WebRTC ecosystem allows various types of CODECs. Some of the common CODECs are G.711 μ-law for North America, G.711 A-law, G.729, and G.722. When two devices that are using two different CODECs communicate with each other, the media gateway translates the CODEC flow between the devices. In other words, a media gateway processes media, and ensures that the end devices are able to communicate with each other.

Push notifications in WebRTC

WebRTC implementations are very common on mobile devices. Unlike web browsers, a mobile device can't keep a websocket connectivity open for a long time. Therefore, it needs to rely on push-notifications from the WebRTC server for all ending requests, such as calls and messages.

Amazon Simple Notification Service (Amazon SNS) lets you send push notifications to apps on mobile devices. These apps could be running on various operating systems such as Apple iOS or Android. The following figure shows a high-level overview of push-notifications flow, from a WebRTC notification server to WebRTC mobile endpoints.

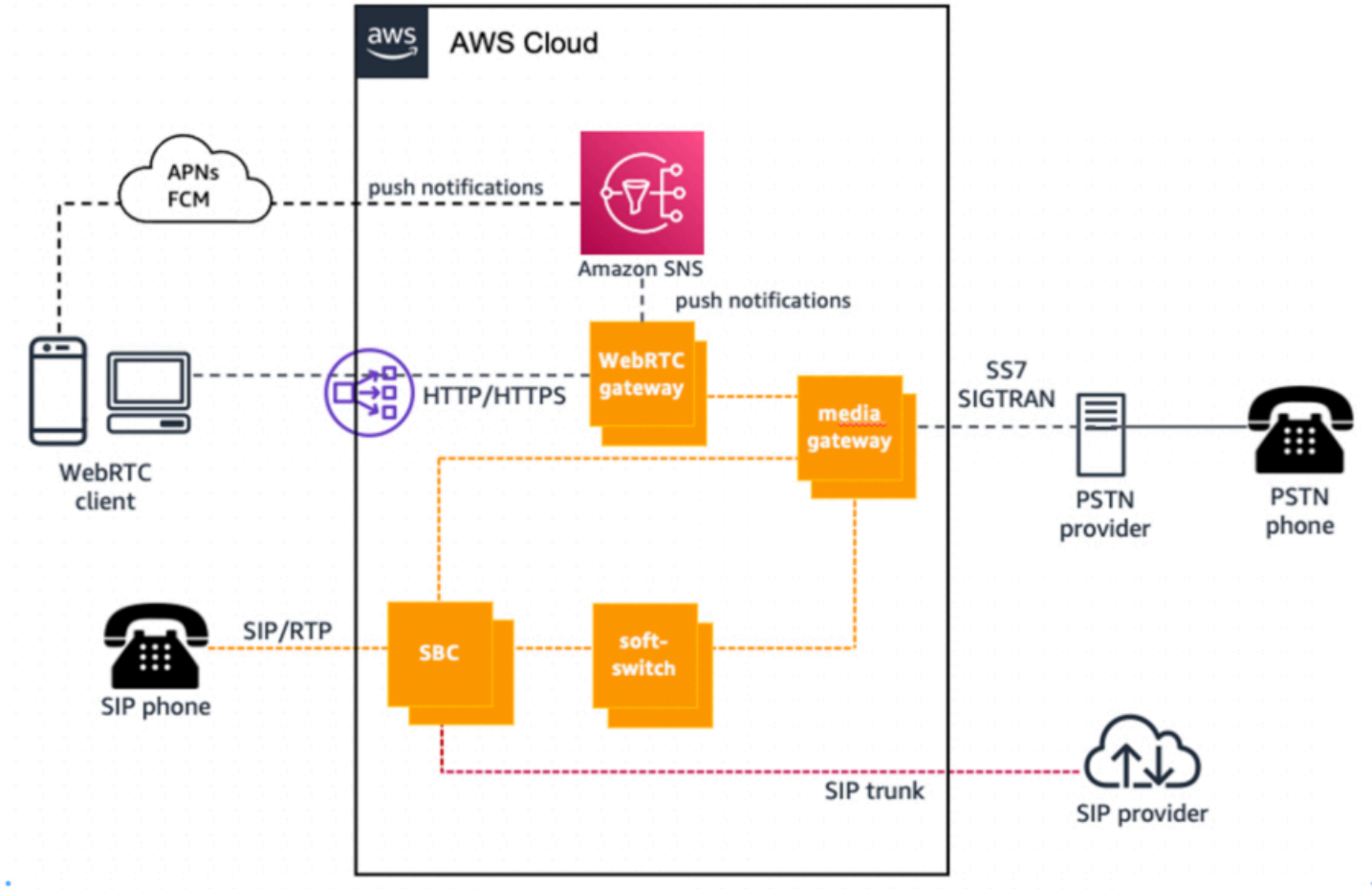


Amazon SNS for push notifications

WebRTC and WebRTC gateway

Web real-time communication (WebRTC) allows you to establish a call from a web browser or request resources from the backend server by using API. The technology is designed with cloud technology in mind and therefore provides various APIs which could be used to establish a call. Because not all of the voice solutions (including SIP) support these APIs, the WebRTC gateway is required to translate API calls into SIP messages and vice versa.

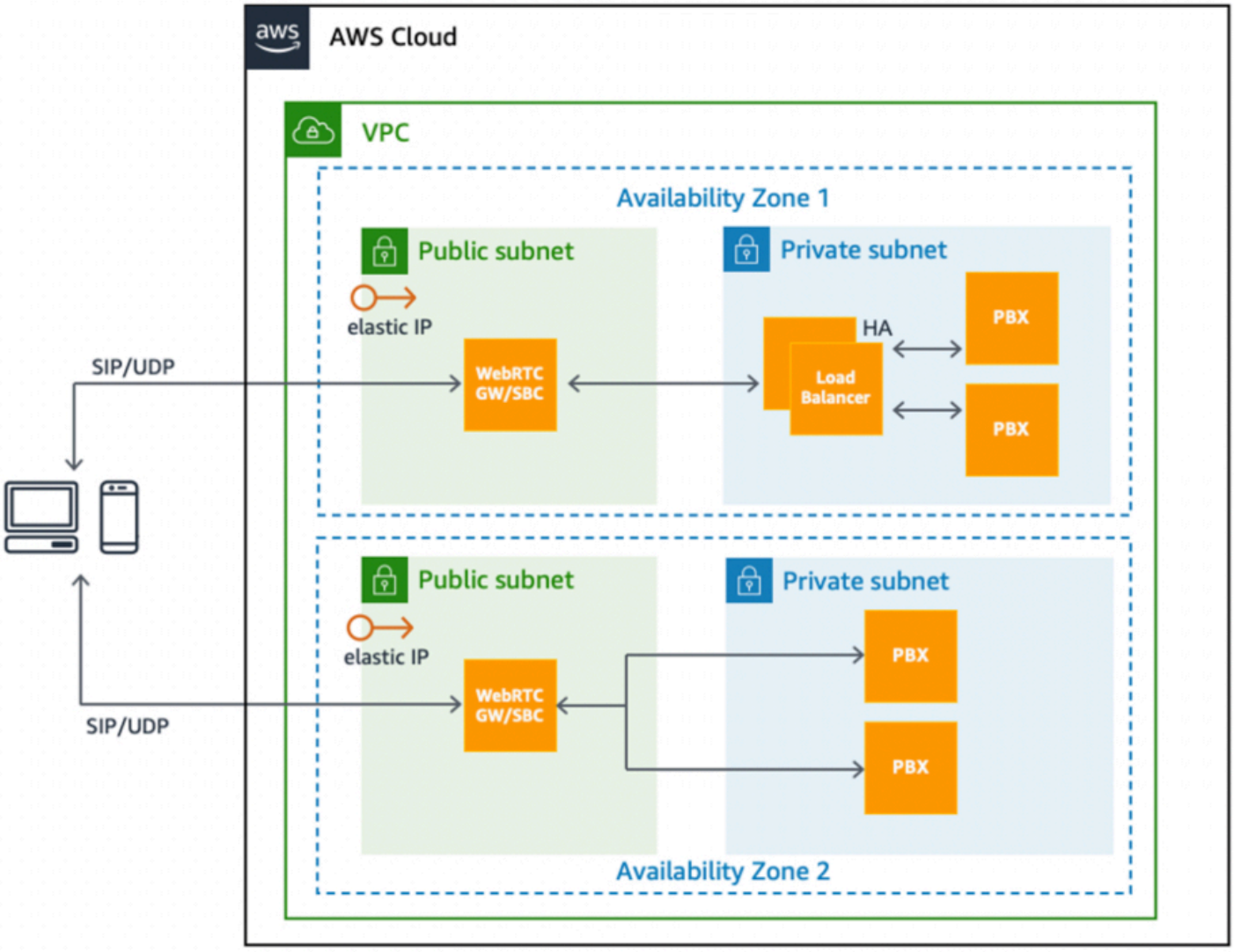
The following figure shows a design pattern for a highly available WebRTC architecture. The incoming traffic from WebRTC clients is balanced by an Application Load Balancer (ALB) with WebRTC running on Amazon Elastic Compute Cloud (Amazon EC2) instances that are part of an Amazon EC2 Auto Scaling group.



A basic topology of an RTC system for voice

Another design pattern for SIP and RTP traffic is to use pairs of SBCs on Amazon EC2 in active-passive mode across Availability Zones, as seen in the following figure. Here, an Elastic IP address

can be dynamically moved between instances upon failure, where the Domain Name Service (DNS) cannot be used.



RTC architecture using Amazon EC2 in a virtual private cloud (VPC)

High availability and scalability on AWS

Most providers of real-time communications align with service levels that provide availability from 99.9% to 99.999%. Depending on the degree of high availability (HA) that you want, you must take increasingly sophisticated measures along the full lifecycle of the application. AWS recommends following these guidelines to achieve a robust degree of high availability:

- Design the system to have no single point of failure. Use automated monitoring, failure detection, and failover mechanisms for both stateless and stateful components
  - Single points of failure (SPOF) are commonly eliminated with an N+1 or 2N redundancy configuration, where N+1 is achieved via load balancing among *active–active* nodes, and 2N is achieved by a pair of nodes in *active–standby* configuration.
  - AWS has several methods for achieving HA through both approaches, such as through a scalable, load balanced cluster or assuming an *active–standby* pair.
- Correctly instrument and test system availability.
- Prepare operating procedures for manual mechanisms to respond to, mitigate, and recover from the failure.

This section focuses on how to achieve no single point of failure using capabilities available on AWS. Specifically, this section describes a subset of core AWS capabilities and design patterns that allow you to build highly available real-time communication applications.

Floating IP pattern for HA between active–standby stateful servers

The floating IP design pattern is a well-known mechanism to achieve automatic failover between an active and standby pair of hardware nodes (media servers). A static secondary virtual IP address is assigned to the active node. Continuous monitoring between the active and standby nodes detects failure. If the active node fails, the monitoring script assigns the virtual IP to the ready standby node and the standby node takes over the primary active function. In this way, the virtual IP floats between the active and standby node.

Applicability in RTC solutions

It is not always possible to have multiple active instances of the same component in service, such as an active–active cluster of N nodes. An active–standby configuration provides the best mechanism for HA. For example, the stateful components in an RTC solution, such as the media server or conferencing server, or even an SBC or database server, are well-suited for an active–standby setup. An SBC or media server has several long running sessions or channels active at a given time, and in the case of the SBC active instance failing, the endpoints can reconnect to the standby node without any client-side configuration due to the floating IP.

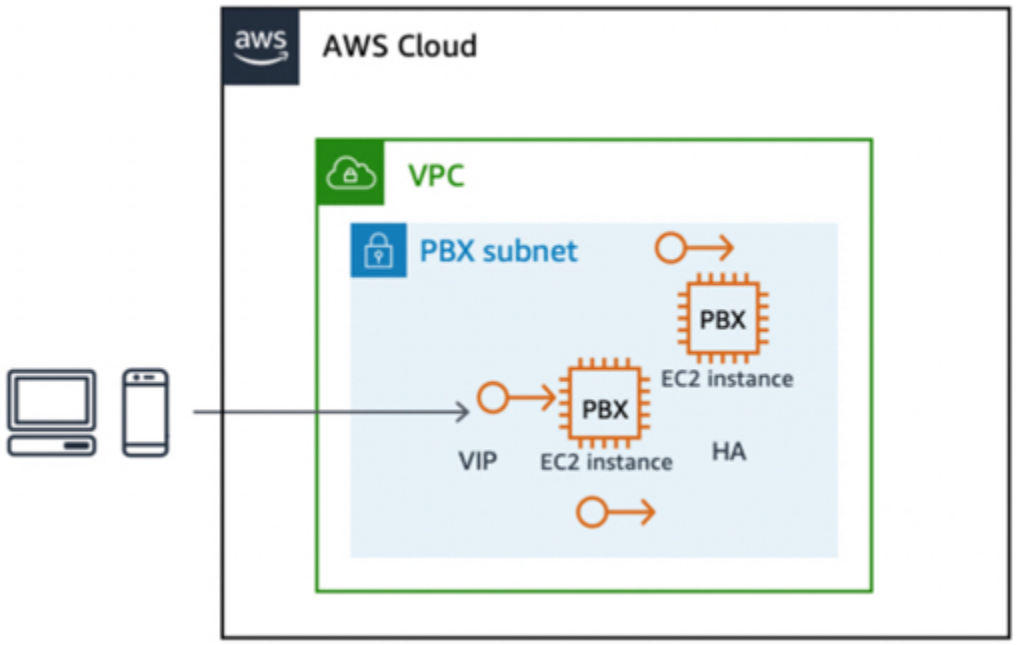
Implementation on AWS

You can implement this pattern on AWS using core capabilities in Amazon Elastic Compute Cloud (Amazon EC2), Amazon EC2 API, Elastic IP addresses, and support on Amazon EC2 for secondary private IP addresses.

To implement the floating IP pattern on AWS:

1. Launch two EC2 instances to assume the roles of primary and secondary nodes, where the primary is assumed to be in *active* state by default.
2. Assign an additional secondary private IP address to the primary EC2 instance.
3. An elastic IP address, which is similar to a virtual IP (VIP), is associated with the secondary private address. This secondary private address is the address that is used by external endpoints to access the application.
4. Some operating system (OS) configuration is required to make the secondary IP address added as an alias to the primary network interface.
5. The application must bind to this elastic IP address. In the case of Asterisk software, you can configure the binding through advanced Asterisk SIP settings.
6. Run a monitoring script—custom, KeepAlive on Linux, Corosync, and so on—on each node to monitor the state of the peer node. In the event, that the current active node fails, the peer detects this failure, and invokes the Amazon EC2 API to reassign the secondary private IP address to itself.

   Therefore, the application that was listening on the VIP associated with the secondary private IP address becomes available to endpoints via the standby node.



Failover between stateful EC2 instances using an elastic IP address

Benefits

This approach is a reliable low-budget solution that protects against failures at the EC2 instance, infrastructure, or application level.

Limitations and extensibility

This design pattern is typically limited to within a single Availability Zone. It can be implemented across two Availability Zones, but with a variation. In this case, the Floating Elastic IP address is re-associated between active and standby node in different Availability Zones via the re-associate elastic IP address API available. In the failover implementation shown in the preceding figure, calls in progress are dropped and endpoints must reconnect. It is possible to extend this implementation with replication of underlying session data to provide seamless failover of sessions or media continuity as well.

Load balancing for scalability and HA with WebRTC and SIP

Load balancing a cluster of active instances based on predefined rules, such as round robin, affinity or latency, and so on, is a design pattern widely popularized by the stateless nature of HTTP requests. In fact, load balancing is a viable option in case of many RTC application components.

The load balancer acts as the reverse proxy or entry point for requests to the desired application, which itself is configured to run in multiple active nodes simultaneously. At any given point in

time, the load balancer directs a user request to one of the active nodes in the defined cluster. Load balancers perform a health check against the nodes in their target cluster and do not send an incoming request to a node that fails the health check. Therefore, a fundamental degree of high availability is achieved by load balancing. Also, because a load balancer performs active and passive health checks against all cluster nodes in sub-second intervals, the time for failover is near instantaneous.

The decision on which node to direct is based on system rules defined in the load balancer, including:

- Round robin
- Session or IP affinity, which ensures that multiple requests within a session or from the same IP are sent to the same node in the cluster
- Latency based
- Load based

Applicability in RTC architectures

The WebRTC protocol makes it possible for WebRTC Gateways to be easily load balanced via an HTTP-based load balancer, such as Elastic Load Balancing (ELB), Application Load Balancer (ALB), or Network Load Balancer (NLB). With most SIP implementations relying on transport over both Transmission Control Protocol (TCP) and User Datagram Protocol (UDP), you need network- or connection-level load balancing with support for both TCP and UDP based traffic is needed.

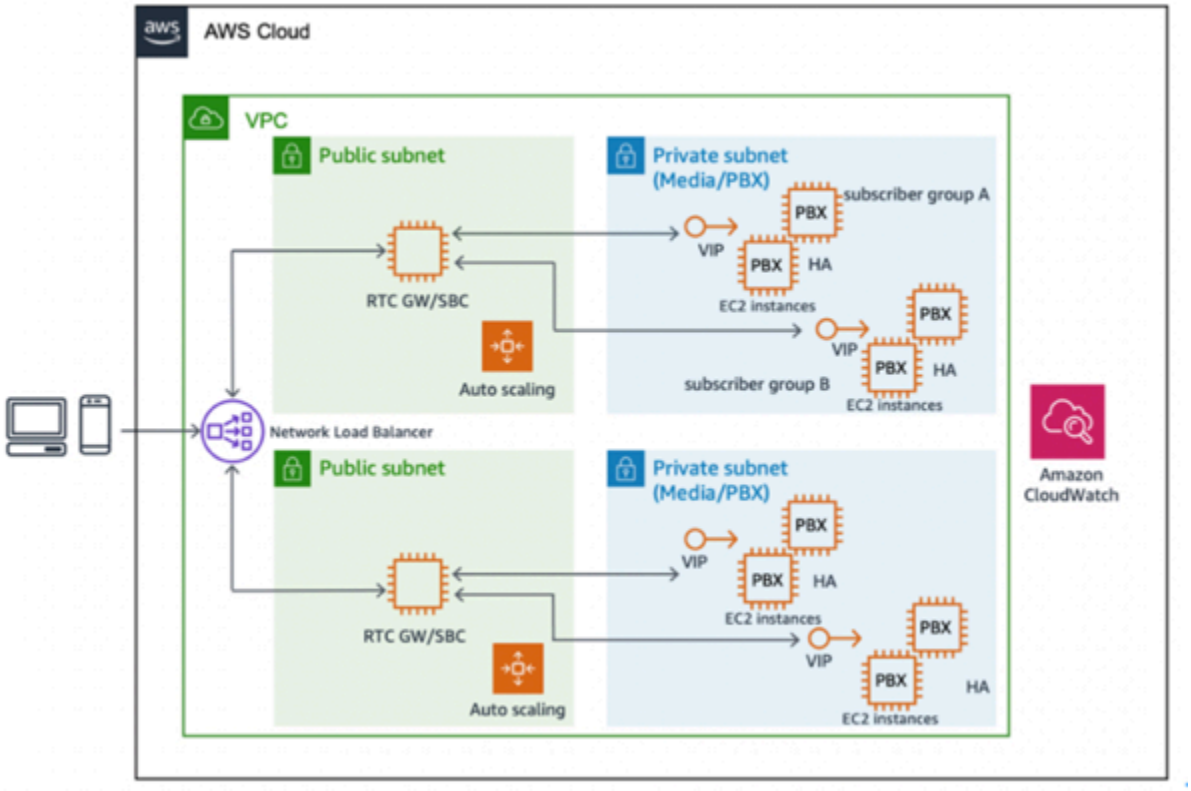
Load balancing on AWS for WebRTC using Application Load Balancer and Auto Scaling

In the case of WebRTC based communications, Elastic Load Balancing provides a fully managed, highly available and scalable load balancer to serve as the entry point for requests, which are then directed to a target cluster of EC2 instances associated with Elastic Load Balancing. Because WebRTC requests are stateless, you can use Amazon EC2 Auto Scaling, to provide fully automated and controllable scalability, elasticity, and high availability.

The Application Load Balancer provides a fully managed load balancing service that is highly available using multiple Availability Zones, and scalable. This supports the load balancing of WebSocket requests that handle the signaling for WebRTC applications and bidirectional

communication between the client and server using a long running TCP connection. The Application Load Balancer also supports content-based routing and sticky sessions, routing requests from the same client to the same target using load balancer generated cookies. If you enable sticky sessions, the same target receives the request and can use the cookie to recover the session context.

The following figure shows the target topology.



WebRTC scalability and high availability architecture

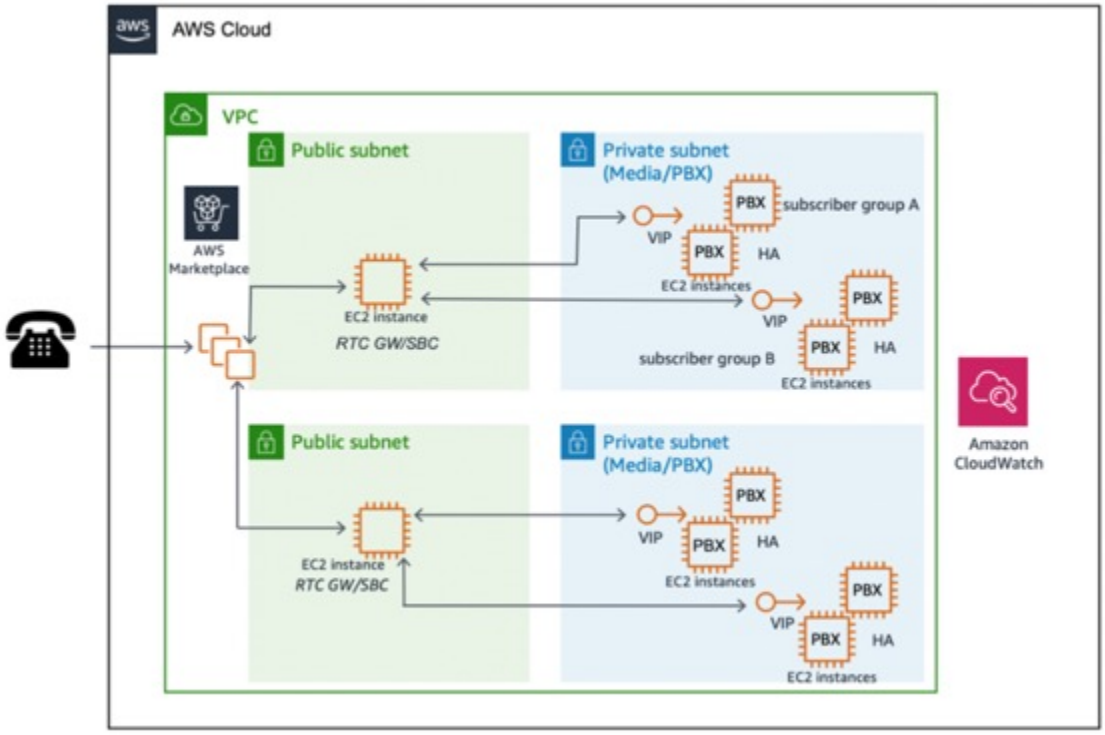
Implementation for SIP using Network Load Balancer or an AWS Marketplace product

In the case of SIP-based communications, the connections are made over TCP or UDP, with the majority of RTC applications using UDP. If SIP/TCP is the signal protocol of choice, then it is feasible to use the Network Load Balancer for fully managed, highly available, scalable and performance load balancing.

A Network Load Balancer operates at the connection level (Layer four), routing connections to targets such as Amazon EC2 instances, containers, and IP addresses based on IP protocol data. Ideal for TCP or UDP traffic load balancing, network load balancing is capable of handling millions

of requests per second while maintaining ultra-low latencies. It is integrated with other popular AWS services, such as Amazon EC2 Auto Scaling, Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS) and AWS CloudFormation.

If SIP connections are initiated, another option is to use AWS Marketplace commercial off-the-shelf software (COTS). The AWS Marketplace offers many products that can handle UDP and other types of layer four connection load balancing. COTS typically include support for high availability and commonly integrate with features, such as Amazon EC2 Auto Scaling, to further enhance availability and scalability. The following figure shows the target topology:



SIP-based RTC scalability with AWS Marketplace product

Cross-Region DNS-based load balancing and failover

Amazon Route 53 provides a global DNS service that can be used as a public or private endpoint for RTC clients to register and connect with media applications. With Amazon Route 53, DNS health checks can be configured to route traffic to healthy endpoints or to independently monitor the health of your application.

The Amazon Route 53 Traffic Flow feature makes it easy for you to manage traffic globally through a variety of routing types, including latency-based routing, geo DNS, geoproximity, and weighted round robin—all of which can be combined with DNS Failover to enable a variety of low-latency,

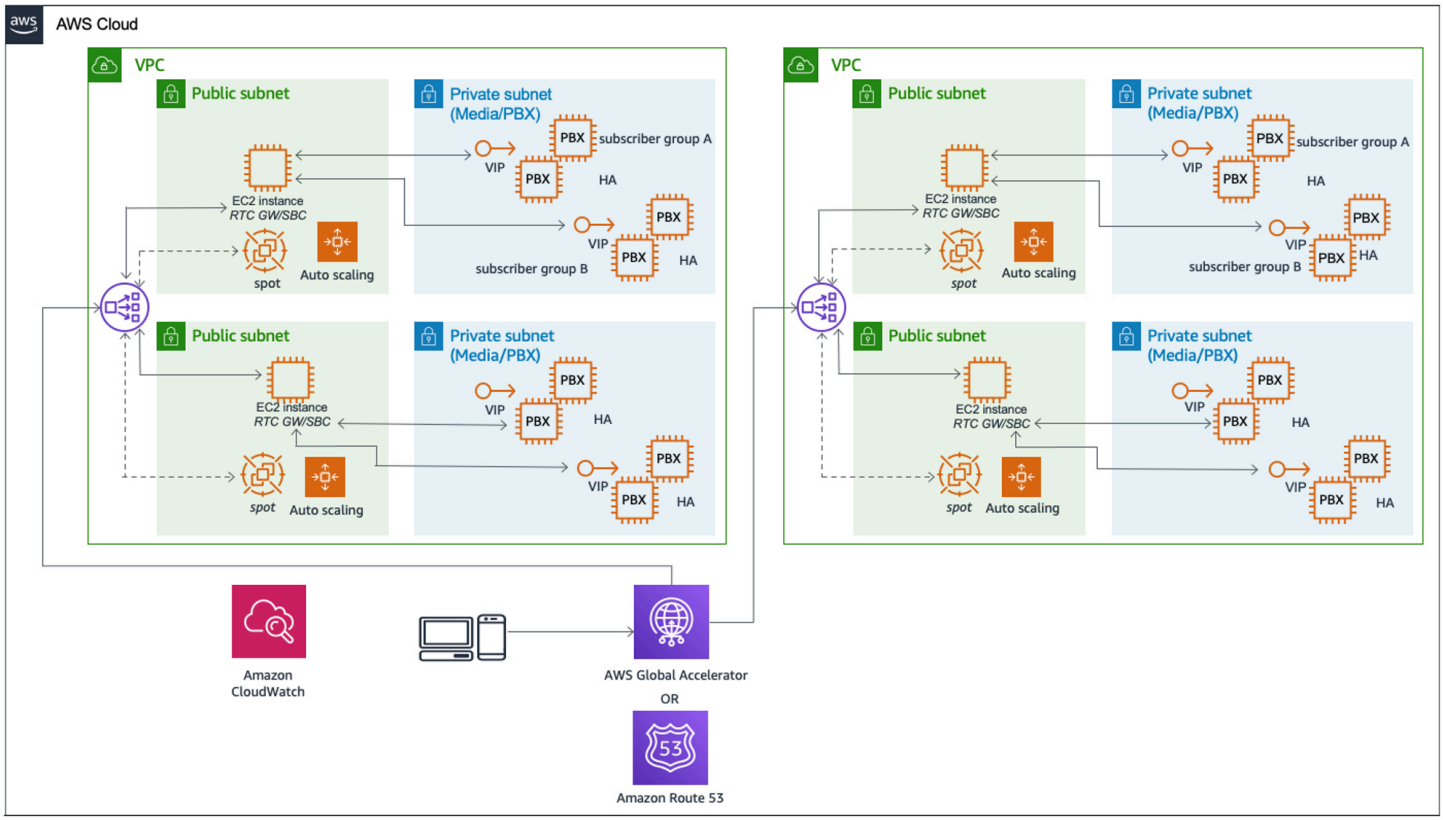
fault-tolerant architectures. The Amazon Route 53 Traffic Flow simple visual editor allows you to manage how your end users are routed to your application's endpoints—whether in a single AWS Region or distributed around the globe.

In the case of global deployments, the latency-based routing policy in Route 53 is especially useful to direct customers to the nearest point of presence for a media server to improve the quality of service associated with real-time media exchanges.

Note that to enforce a failover to a new DNS address, client caches must be flushed. Also, DNS changes may have a lag as they are propagated across global DNS servers. You can manage the refresh interval for DNS lookups with the Time to Live attribute. This attribute is configurable at the time of setting up DNS policies.

To reach global users quickly or to meet the requirements of using a single public IP, AWS Global Accelerator can also be used for cross-Region failover. AWS Global Accelerator is a networking service that improves availability and performance for applications with both local and global reach. AWS Global Accelerator provides static IP addresses that act as a fixed entry point to your application endpoints, such as your Application Load Balancers, Network Load Balancers, or Amazon EC2 instances in single or multiple AWS Regions. It uses the AWS global network to optimize the path from your users to your applications, improving performance, such as the latency of your TCP and UDP traffic.

AWS Global Accelerator continually monitors the health of your application endpoints, and automatically redirects traffic to the nearest healthy endpoints in the event of current endpoints turning unhealthy. For additional security requirements, Accelerated Site-to-Site VPN uses AWS Global Accelerator to improve the performance of VPN connections by intelligently routing traffic through the AWS Global Network and AWS edge locations.



Inter-Region high availability design using AWS Global Accelerator or Amazon Route 53

Data durability and HA with persistent storage

Most RTC applications rely on persistent storage to store and access data for authentication, authorization, accounting (session data, call detail records, etc.), operational monitoring, and logging. In a traditional data center, ensuring high availability and durability for the persistent storage components (databases, file systems, and so on) typically requires heavy lifting via the setup of a storage area network (SAN), Redundant Array of Independent Disks (RAID) design, and processes for backup, restore, and failover processing. The AWS Cloud greatly simplifies and enhances traditional data center practices around data durability and availability.

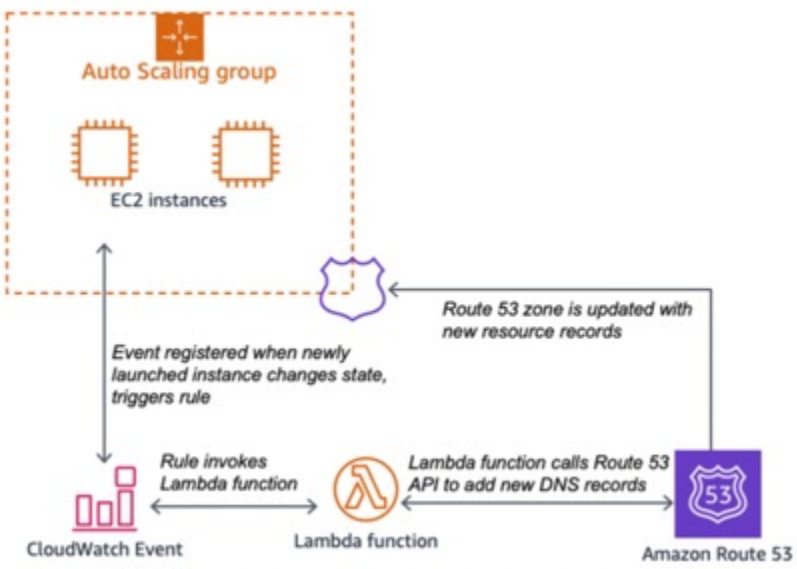
For object storage and file storage, AWS services like Amazon Simple Storage Service (Amazon S3) and Amazon Elastic File System (Amazon EFS) provide managed high availability and scalability. Amazon S3 has a data durability of 99.999999999% (11 nines).

For transactional data storage, customers have the option to take advantage of the fully managed Amazon Relational Database Service (Amazon RDS) that supports Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle, and Microsoft SQL Server with high availability deployments. For the

registrar function, subscriber profile, or accounting records storage (such as CDRs), the Amazon RDS provides a fault-tolerant, highly available and scalable option.

Dynamic scaling with AWS Lambda, Amazon Route 53, and Amazon EC2 Auto Scaling

AWS allows the chaining of features and the ability to incorporate custom serverless functions as a service based on infrastructure events. One such design pattern that has many versatile uses in RTC applications is the combination of automatic scaling lifecycle hooks with Amazon CloudWatch Events, Amazon Route 53, and AWS Lambda functions. AWS Lambda functions can embed any action or logic. The following figure demonstrates how these features chained together can enhance system reliability and scalability with automation.



Automatic scaling with dynamic updates to Amazon Route 53

Highly available WebRTC with Amazon Kinesis Video Streams

Amazon Kinesis Video Streams offers real-time media streaming via WebRTC, allowing users to capture, process, and store media streams for playback, analytics, and machine learning. These streams are highly available, scalable, and compliant with WebRTC standards. Amazon Kinesis Video Streams include a WebRTC signaling endpoint for fast peer discovery and secure connection establishment. It includes managed Session Traversal Utilities for NAT (STUN) and Traversal Using Relays around NAT (TURN) end-points for real-time exchange of media between peers. It also includes a free open-source SDK that directly integrates with camera firmware to enable secure

communication with Amazon Kinesis Video Streams end-points, allowing for peer discovery and media streaming. Finally, it provides client libraries for Android, iOS, and JavaScript that allow WebRTC compliant mobile and web players to securely discover and connect with a camera device for media streaming and two-way communication.

Highly available SIP trunking with Amazon Chime Voice Connector

Amazon Chime Voice Connector delivers a pay-as-you-go SIP trunking service that enables companies to make and/or receive secure and inexpensive phone calls with their phone systems. Amazon Chime Voice Connector is a low-cost alternative to service provider SIP trunks or Integrated Services Digital Network (ISDN) Primary Rate Interfaces (PRIs). Customers have the option to enable inbound calling, outbound calling, or both.

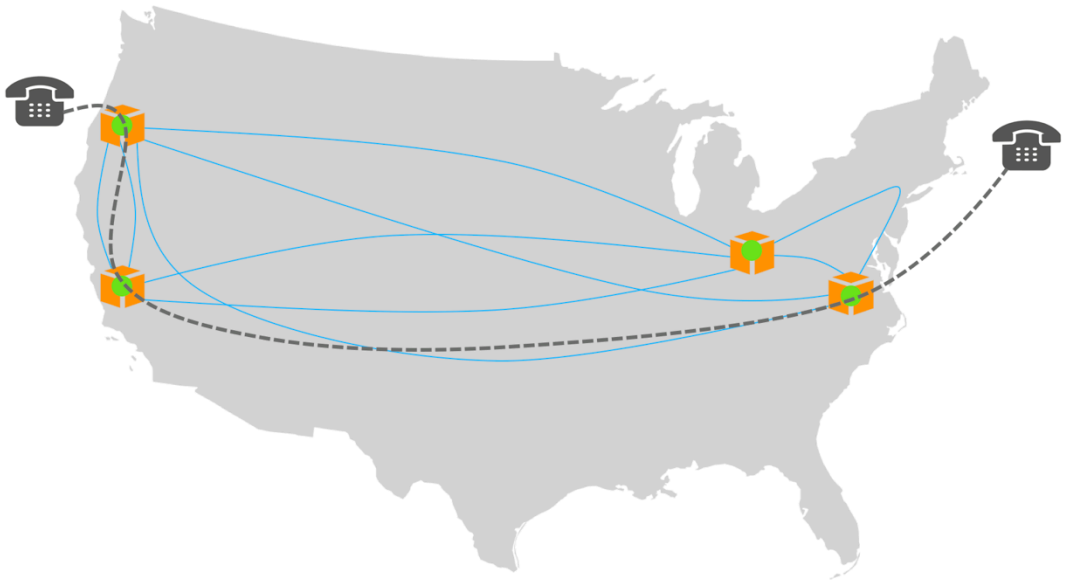
The service uses the AWS network to deliver a highly available calling experience across multiple AWS Regions. You can stream audio from SIP trunking telephone calls, or forwarded SIP-based media recording (SIPREC) feeds to Amazon Kinesis Video Streams to gain insights from business calls in real time. You can quickly build applications for audio analytics through integration with Amazon Transcribe and other common machine learning libraries.

Best practices from the field

This section summarizes the best practices that have been implemented by some of largest and most successful AWS customers that run large real-time Session Initiation Protocol (SIP) workloads. AWS customers wanting to run their own SIP infrastructure in the public cloud would find these best practices valuable as they can help increase the reliability and resiliency of the system in case of different kinds of failures. Although some of these best practices are SIP specific, most of them are applicable to any real-time communication application running on AWS.

Create a SIP overlay

AWS has a robust, scalable and redundant network backbone that provides connectivity between different AWS Regions. When a network event, such as a fiber cut, degrades an AWS backbone link, traffic is quickly failed over to redundant paths using network level routing protocols, such as Border Gateway Protocol (BGP). This network level traffic engineering is a black box to AWS customers and most do not even notice these failover events. However, customers that run real-time workloads, such as voice, high quality video, and low latency messaging, do sometimes notice these events. So, how can an AWS customer implement their own traffic engineering on top of what is provided by AWS at the network level? The solution is deploying SIP infrastructure at many different AWS Regions. As part of the call control features, SIP also provides the ability to route calls through specific SIP proxies.

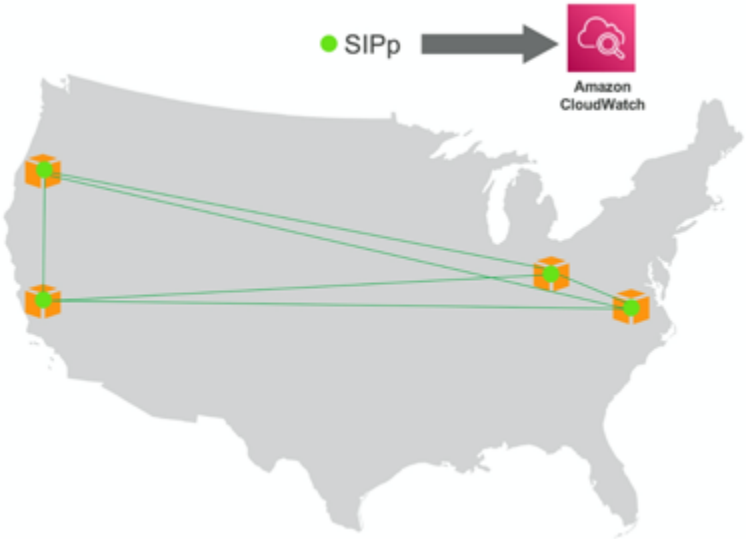


Using SIP routing to override network routing

In the preceding figure, SIP infrastructure (represented by green dots inside the cubes) is running in all four US Regions. The solid blue lines represent a fictional depiction of the AWS backbone. If no SIP routing is implemented, a call originating in the US west coast and destined for the US east coast goes over the backbone link that is directly connecting the Oregon and Virginia regions. The diagram shows how a customer might override the network level routing and make the same call between Oregon and Virginia routed through California using SIP routing. This type of SIP traffic engineering can be implemented using SIP proxies and media gateways based on network metrics such as SIP retransmissions and customer specific business preferences.

Perform detailed monitoring

End users of real-time voice and video applications expect the same level of performance as they achieve with traditional telephony services. So, when they experience issues with an application, it ends up hurting the provider's reputation. To be proactive rather than reactive, it is imperative that detailed monitoring be deployed at every part of the system that serves end users.



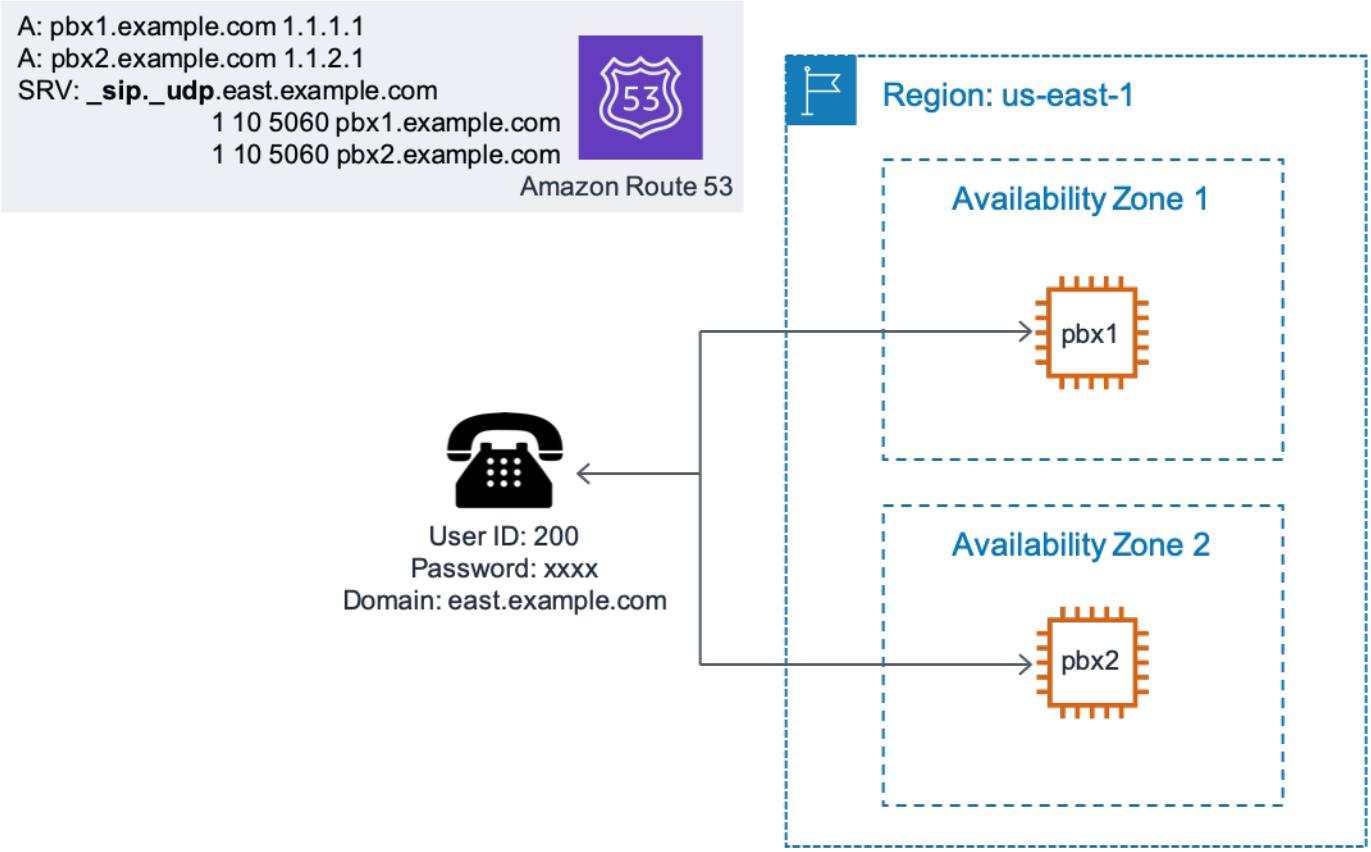
Using SIPp to monitor VoIP infrastructure

Many open source tools, such as iPerf or SIPp, and VOIPMonitor, are available to use in monitoring SIP/RTP traffic. In the preceding example, nodes running SIP in client and server modes are measuring SIP metrics such as Successful Calls and SIP Retransmits between all four US AWS Regions. These metrics can then be exported into Amazon CloudWatch using a custom script. Using CloudWatch, customers can create alarms on these custom metrics based on a certain threshold value. Automatic or manual remediation actions can then be taken based on the state of these CloudWatch alarms.

For customers not wanting to allocate engineering resources needed to develop and maintain a custom monitoring system, many good VoIP monitoring solutions are available on the market, such as ThousandEyes. An example of a remediation action is changing the SIP routing based on increased SIP retransmits.

Use DNS for load balancing and floating IPs for failover

IP telephony clients that support DNS SRV capability can efficiently use the redundancy built into the infrastructure by load balancing clients to different SBCs/PBXs.



Using DNS SRV records to load balance SIP clients

The preceding figure shows how customers can use the SRV records to load balance SIP traffic. Any IP telephony client that supports the SRV standard will look for the sip. `<transport protocol>` prefix in an SRV type DNS record. In the example, the answer section from DNS contains both of the PBXs running in different AWS Availability Zones. However, in addition to the endpoint URIs, the SRV record contains three additional pieces of information:

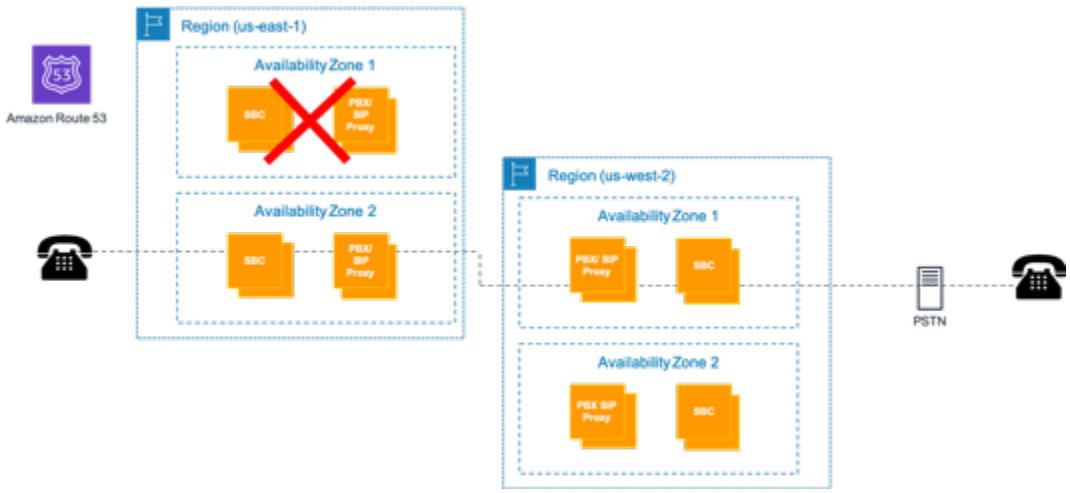
- The first number is the **Priority** (1 in the example above). A lower priority is preferred over higher.
- The second number is the **Weight** (10 in the example above).
- And the third number is the **Port** to be used (5060).

Since the priority is the same (1) for both PBXs servers, the clients use the weight to load balance between the two PBXs. In this case, since the weights are the same, SIP traffic should be load balanced equally between the two PBXs.

DNS can be a good solution for client load balancing, but what about implementing failover by changing/updating DNS 'A' records? This method is discouraged because of inconsistency found in DNS caching behavior within the client and intermediate nodes. A better approach for intra-AZ failover between a cluster of SIP nodes is to use the EC2 IP reassignment where an impaired host's IP address is instantly reassigned to a healthy host by using the EC2 API. Paired with a detailed monitoring and health check solution, IP reassignment of a failed node ensures that traffic is moved over to a healthy host in a timely manner that minimizes end user disruption.

Use multiple Availability Zones

Each AWS Region is subdivided into separate Availability Zones. Each Availability Zone has its own power, cooling, and network connectivity and thus forms an isolated failure domain. Within the constructs of AWS, customers are encouraged to run their workloads in more than one Availability Zone. This ensures that customer applications can withstand even a complete Availability Zone failure - a very rare event in itself. This recommendation stands for real-time SIP infrastructure as well.



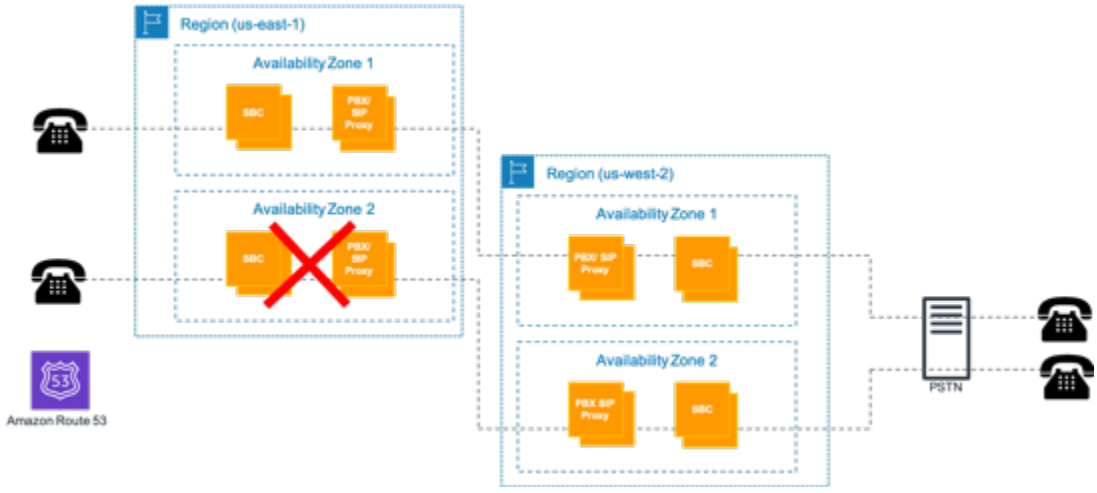
Handling Availability Zone failure

Suppose a catastrophic event (such as category five hurricane) causes a complete Availability Zone outage in the us-east-1 Region. With the infrastructure running as shown in the diagram, all SIP clients that were originally registered with the nodes in the failed Availability Zone should re-register with the SIP nodes running in Availability Zone #2. (Test this behavior with your SIP clients/phones to make sure it is supported.) Although the active SIP calls at the time of the Availability Zone outage are lost, any new calls are routed through Availability Zone 2.

To summarize, DNS SRV records should point the client to multiple 'A' records, one in each Availability Zone. Each of those 'A' records should, in turn, point to multiple IP addresses of SBCs/PBXs in that Availability Zone providing both intra- and inter-Availability Zone resiliency. Both intra- and inter-Availability Zone failover can be implemented by using IP reassignment if the IPs are public. Private IPs, however, cannot be reassigned across Availability Zones. If a customer is using private IP addressing, then they would have to rely on the SIP clients re-registering with the backup SBC/PBX for inter-Availability Zone failover.

Keep traffic within one Availability Zone and use EC2 placement groups

Also known as Availability Zone Affinity, this best practice also applies to the rare event of a complete Availability Zone failure. It is recommended that you eliminate any cross-AZ traffic such that any SIP or RTP traffic that enters one Availability Zone should remain in that Availability Zone until it exits the Region.



Availability Zone affinity (at most, 50% of active calls are lost)

The preceding figure shows a simplified architecture that uses Availability Zone affinity. The comparative advantage of this approach becomes clear if one accounts for the effects of a complete Availability Zone outage. As depicted in the diagram, if Availability Zone 2 is lost, 50% of active calls are affected at most (assuming equal load balancing between Availability Zones). Had Availability Zone Affinity not been implemented, some calls would flow between Availability Zones in one Region and a failure would most likely affect more than 50% of active calls.

To minimize latency for traffic, AWS also recommends that you consider using EC2 placement groups within each Availability Zone. Instances launched within the same EC2 placement group have higher bandwidth and reduced latency as EC2 ensures network proximity of these instances relative to each other.

Use enhanced networking EC2 instance types

Choosing the right instance type on Amazon EC2 ensures system reliability as well as efficient usage of infrastructure. EC2 provides a wide selection of instance types optimized to fit different use cases. Instance types comprise varying combinations of CPU, memory, storage, and networking capacity and give you the flexibility to choose the appropriate mix of resources for your applications. These enhanced networking instance types ensure that the SIP workloads running on them have access to consistent bandwidth and comparatively lower aggregate latency. A recent addition to Amazon EC2 is the availability of the Elastic Network Adapter (ENA) that provides up to 100 Gbps of bandwidth. The latest catalog of EC2 instance types and associated features can be found on the EC2 instance types page.

For most customers, the latest generation of Compute Optimized instances should provide the best value for the cost. For example, the C5N supports the new Elastic Network Adapter with bandwidth up to 100 Gbps with millions of packets per second (PPS). Most real-time applications would also benefit from using the Intel Data Plane Developer Kit (DPDK) which can greatly boost network packet processing.

However, it is always a best practice to benchmark the various EC2 instance types according to your requirements to see which instance type works best for you. Benchmarking also enables you to find other configuration parameters, such as the maximum number of calls a certain instance type can process at a time.

Security considerations

RTC application components typically run directly on internet facing Amazon EC2 instances. In addition to TCP, flows use protocols like UDP and SIP. In these cases, AWS Shield Standard protects Amazon EC2 instances from common infrastructure layer (Layer 3 and 4) DDoS attacks, such as UDP reflection attacks, DNS reflection, NTP reflection, SSDP reflection, and so on. AWS Shield Standard uses various techniques like priority-based traffic shaping that are automatically engaged when a well-defined DDoS attack signature is detected.

AWS also provides advanced protection against large and sophisticated DDoS attacks for these applications by enabling AWS Shield Advanced on Elastic IP addresses. AWS Shield Advanced provides enhanced DDoS detection that automatically detects the type of AWS resource and size of EC2 instance and applies appropriate predefined mitigations with protections against SYN or UDP floods. With AWS Shield Advanced, customers can also create their own custom mitigation profiles by engaging the 24x7 AWS DDoS Response Team (DRT). AWS Shield Advanced also ensures that during a DDoS attack, all of your Amazon VPC Network Access Control Lists (ACLs) are automatically enforced at the border of the AWS network providing you with access to additional bandwidth and scrubbing capacity to mitigate large volumetric DDoS attacks.

Conclusion

Real-time communication (RTC) workloads can be deployed on AWS to attain scalability, elasticity, and high availability while meeting the key requirements. Today, several customers are using AWS, its partners, and open source solutions to run RTC workloads with reduced cost and faster agility as well as a reduced global footprint.

The reference architectures and best practices provided in this white paper can help customers successfully set up RTC workloads on AWS and optimize the solutions to meet end user requirements while optimizing for the cloud.

Acronyms

Acronyms used in this document include:

ACL — Access Control List

ALB — Application Load Balancer

APNs — Apple Push Notification service

BGP — Border Gateway Protocol

CDR — Call Detail Records

COTS — commercial off-the-shelf software

DDoS — distributed denial-of-service

DNS — Domain Name System

DPDK — Intel Data Plane Developer Kit

DRT — DDoS Response Team

ENA — Elastic Network Adapter

EPC – Evolved Packet Core

FCM — Firebase Cloud Messaging

HA — High Availability

IRC — Internet Relay Chat

ISDN — Integrated Services Digital Network

NAT — network address translation

OPUS — online positioning user support

PBX — Private Branch Exchange

PRI — Primary Rate Interface

PSTN — Public Switched Telephone Network

RAID — Redundant Array of Independent Disks

RTC — real-time communication

RTP —Real-time Transport Protocol

SAN — Storage Area Network

SBC — session border controller

SIP —Session Initiation Protocol

SPOF — single points of failure

SRV — Service

SS7 — Signaling System n.7

STUN — Session Traversal Utilities for NAT

SYN —Synchronize

TCP — Transmission Control Protocol

TDM — time division multiplexing

TURN — Traversal Using Relays around NAT

UDP — User Datagram Protocol

URI — Uniform Resource Identifiers

VIP — virtual IP

VNF — Virtual Network Function

VoIP — Voice Over IP

VPC — Virtual Private Cloud

WebRTC — web real-time communication

Contributors

The following individuals and organizations contributed to this document:

- Mounir Chennana, Senior Solutions Architect, Amazon Web Services
- Mohammed Al-Mehdar, Senior Solutions Architect, Amazon Web Services
- Ejaz Sial, Senior Solutions Architect, Amazon Web Services
- Ahmad Khan, Senior Solutions Architect, Amazon Web Services
- Tipu Qureshi, Principal Engineer, AWS Support, Amazon Web Services
- Hasan Khan, Senior Technical Account Manager, Amazon Web Services
- Shoma Chakravarty, WW Technical Leader, Telecom, Amazon Web Services

Document revisions

To be notified about updates to this whitepaper, subscribe to the RSS feed.

Change	Description	Date
Whitepaper updated	Updated for latest services and features.	May 5, 2022
Whitepaper updated	Updated for latest services and features.	February 13, 2020
Initial publication	Whitepaper first published.	October 1, 2018

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

AWS Glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.