Comprender WebSocket Connection para Finesse

Contenido

Introducción

Este documento describe completamente la conexión WebSocket para que, durante la resolución de problemas, los procesos subyacentes se entiendan a fondo.

Prerequisites

Requirements

No hay requisitos específicos para este documento.

Componentes Usado

La información que contiene este documento se basa en las siguientes versiones de software y hardware.

- Cisco Finesse
- UCCX

La información que contiene este documento se creó a partir de los dispositivos en un ambiente de laboratorio específico. Todos los dispositivos que se utilizan en este documento se pusieron en funcionamiento con una configuración verificada (predeterminada). Si tiene una red en vivo, asegúrese de entender el posible impacto de cualquier comando.

Antecedentes

Web Socket es una conexión persistente entre el cliente y el servidor.

Socket web

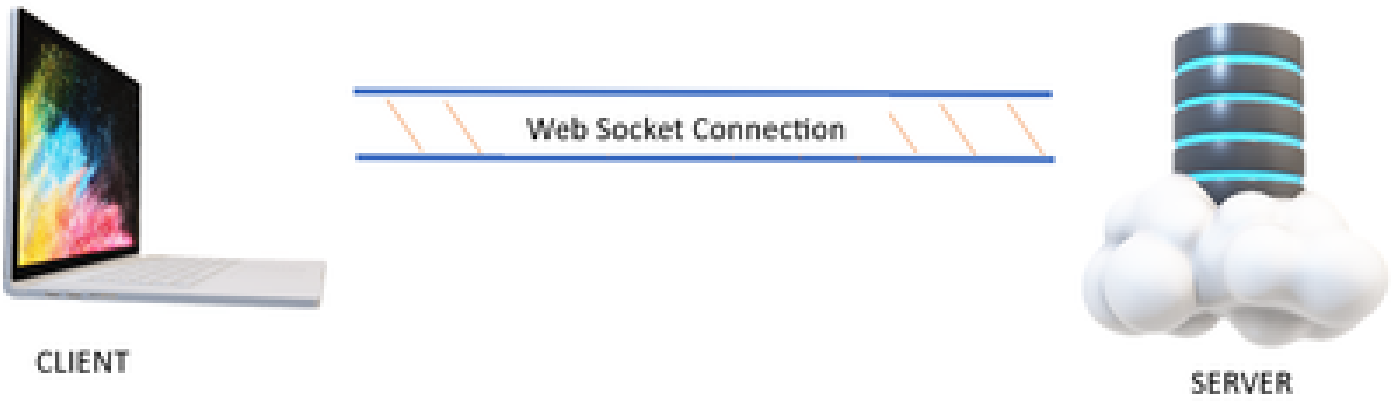
¿Qué se entiende por el término conexión persistente?

Esto significa que una vez establecida la conexión entre el cliente y el servidor, el cliente y el servidor pueden enviar y/o recibir datos en cualquier momento.

Esta es una conexión dúplex completo bidireccional.

El servidor no tiene que esperar a que la solicitud del cliente devuelva los datos.

Del mismo modo, el cliente tampoco tiene que crear una nueva conexión cada vez para enviar datos nuevos al servidor.

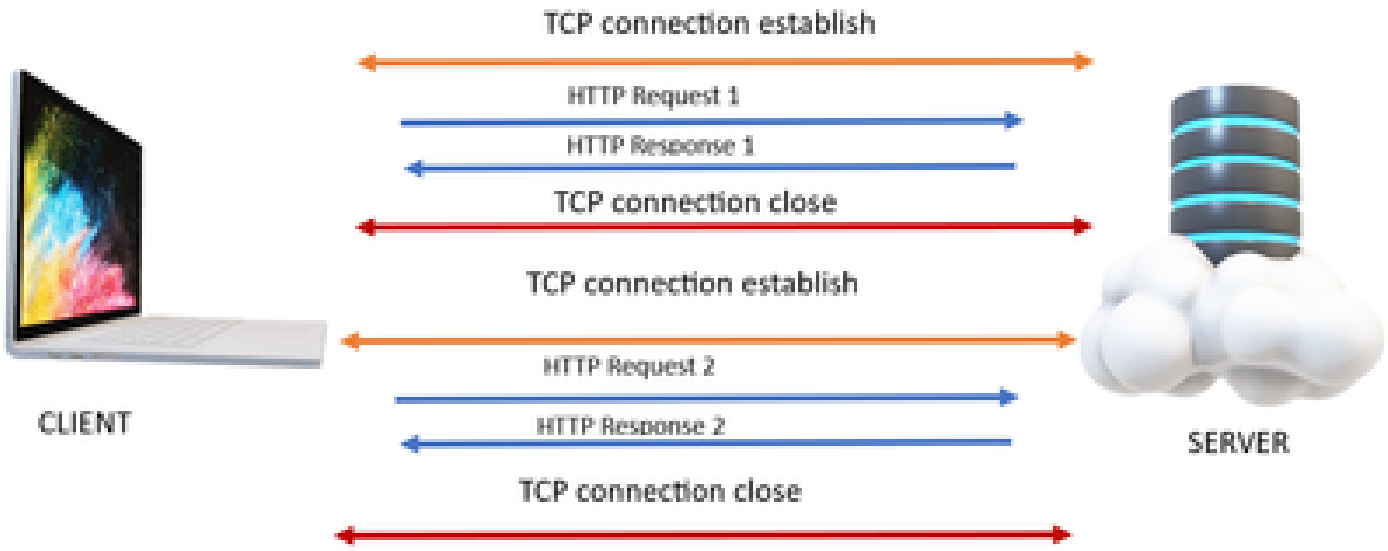


La conexión de socket web se utiliza principalmente en las aplicaciones en las que se requieren actualizaciones de datos en tiempo real.

Por ejemplo, aplicaciones de negociación de acciones, aplicaciones de mensajería y, en nuestro caso, Cisco Finesse.

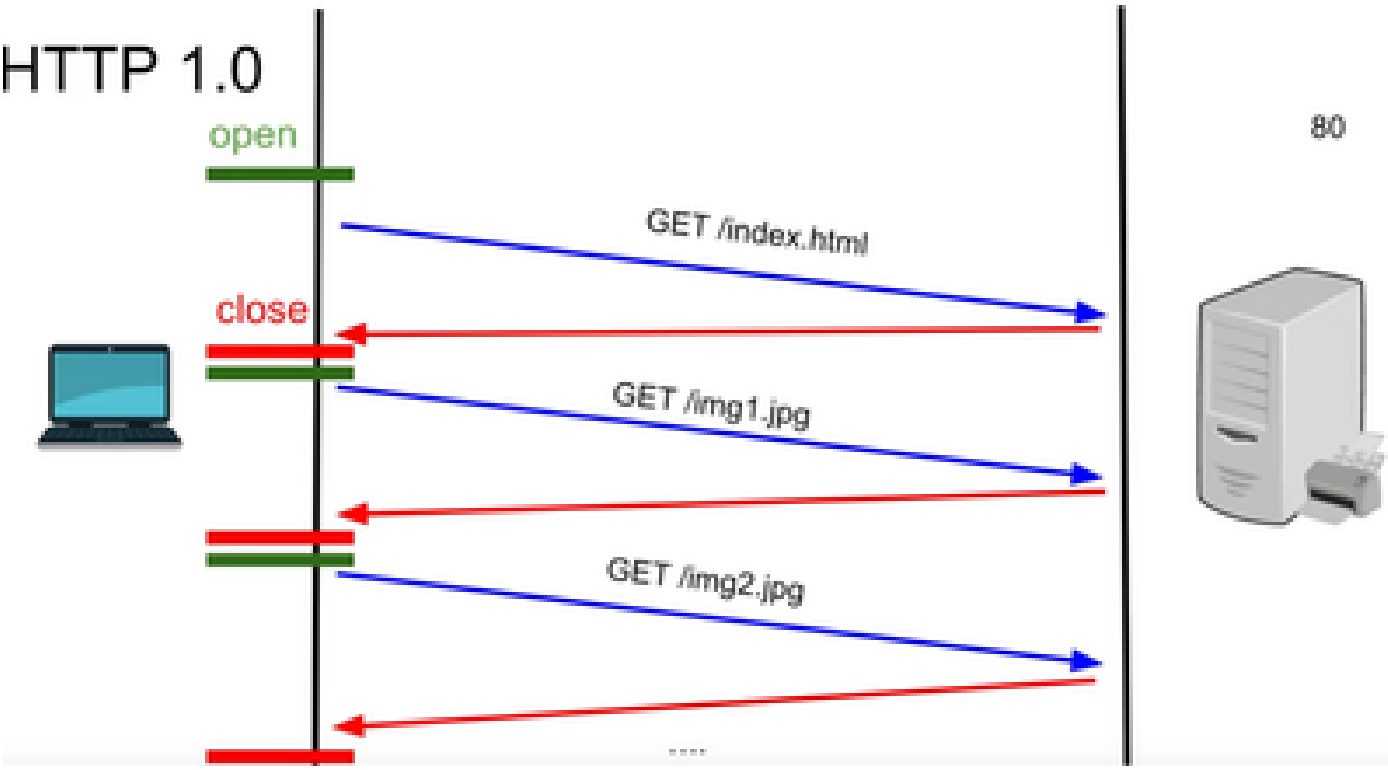
¿Cómo funciona WebSockets?

Tenga en cuenta:

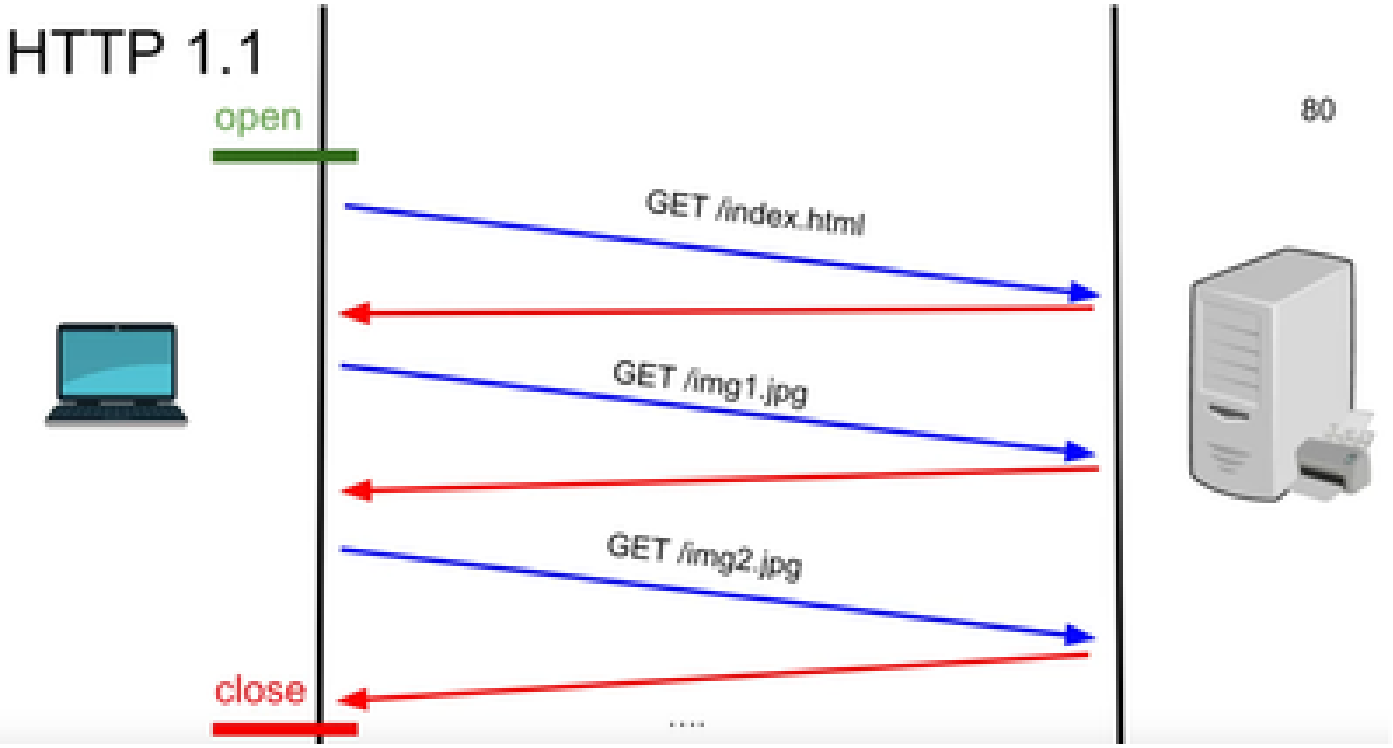


HTTP

1. La conexión TCP (protocolo de enlace de 3 vías) tiene lugar.
2. A continuación, el cliente envía una solicitud HTTP.
3. El servidor envía una respuesta HTTP.
4. Después de un ciclo de respuesta a la solicitud, la conexión TCP se cierra.
5. Para una nueva solicitud HTTP, de nuevo, la conexión TCP se establece primero.



HTTP 1.0: después de cada respuesta de solicitud, el protocolo de enlace TCP se inicia de nuevo para otra respuesta de solicitud HTTP.



HTTP 1.1: Esta conexión funcionó porque se podían enviar y recibir datos y cerrar la conexión.

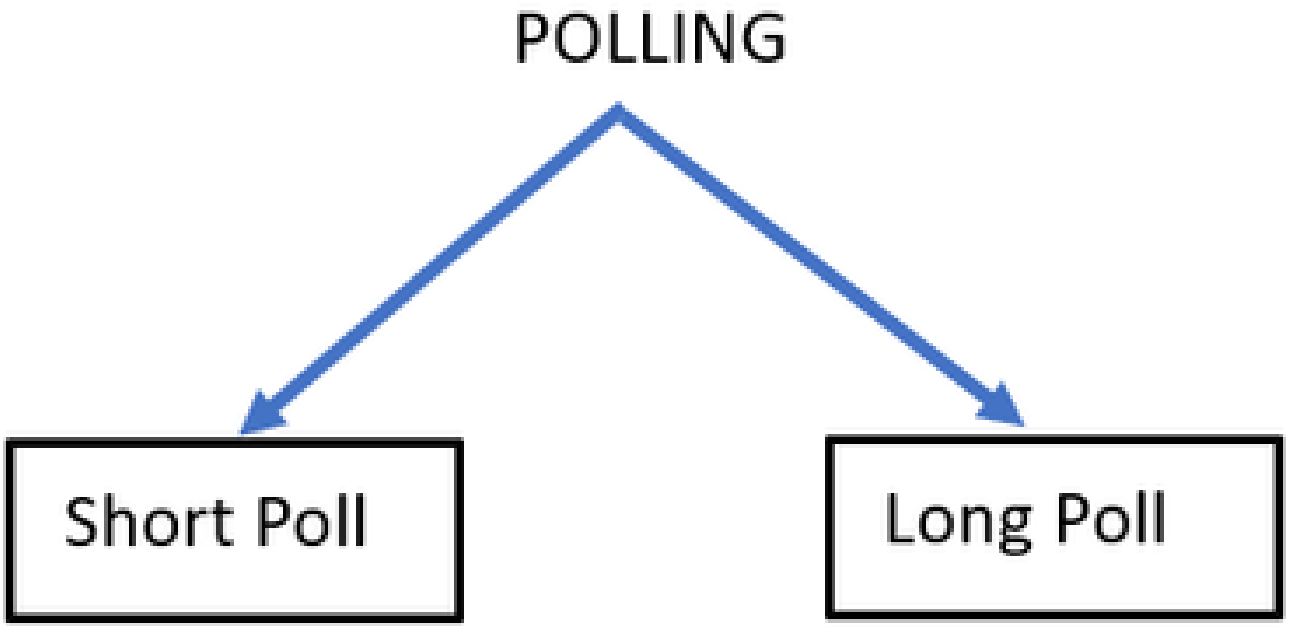
De nuevo, esto no era adecuado para aplicaciones en tiempo real porque el servidor puede enviar algunos datos incluso cuando el cliente no lo solicita. Por lo tanto, este modelo no es eficaz.

Problema con HTTP

El problema comienza con los sistemas en tiempo real.

Para un sitio web que requiere actualizaciones en tiempo real, es muy difícil enviar solicitudes HTTP cada vez para obtener una actualización del servidor y utiliza mucho ancho de banda y causa sobrecarga.

Para resolver esto, se utiliza un mecanismo de HTTP llamado Sondeo.



Encuesta corta - El se implementa cuando se establece un temporizador corto fijo para las solicitudes y respuestas. Por ejemplo, 50 segundos o 1 segundo dependiendo de la implementación.

Si no hay ninguna actualización del otro lado, puede obtener respuestas vacías en ese período de tiempo que pueden desperdiciar recursos.

Encuesta larga - De alguna manera supera la encuesta corta, pero todavía tiene un tiempo fijo para esperar una respuesta.

Si no hay respuesta en ese período de tiempo que es relativamente más largo que el sondeo corto pero aún así es fijo, entonces nuevamente solicite tiempos de espera.

Por lo tanto, la encuesta no es la mejor manera de superar este problema.

Para esto, el otro método a utilizar se llama SSE.

SSE

Eventos enviados por servidor

En este caso, existe una conexión unidireccional entre el servidor y el cliente mediante la cual el servidor puede enviar los datos al cliente en cualquier momento.

Lo que hay que tener en cuenta es que se trata de una conexión unidireccional, lo que significa que solo el servidor puede enviar los datos al cliente y no al revés.

Un ejemplo de caso práctico es el siguiente: Notificaciones o actualizaciones masivas de un servidor a un cliente. Por ejemplo, noticias, actualizaciones en vivo, Instagram en vivo, etc.

Esto no es muy eficaz para las aplicaciones que implican actualizaciones y mensajería en tiempo real.

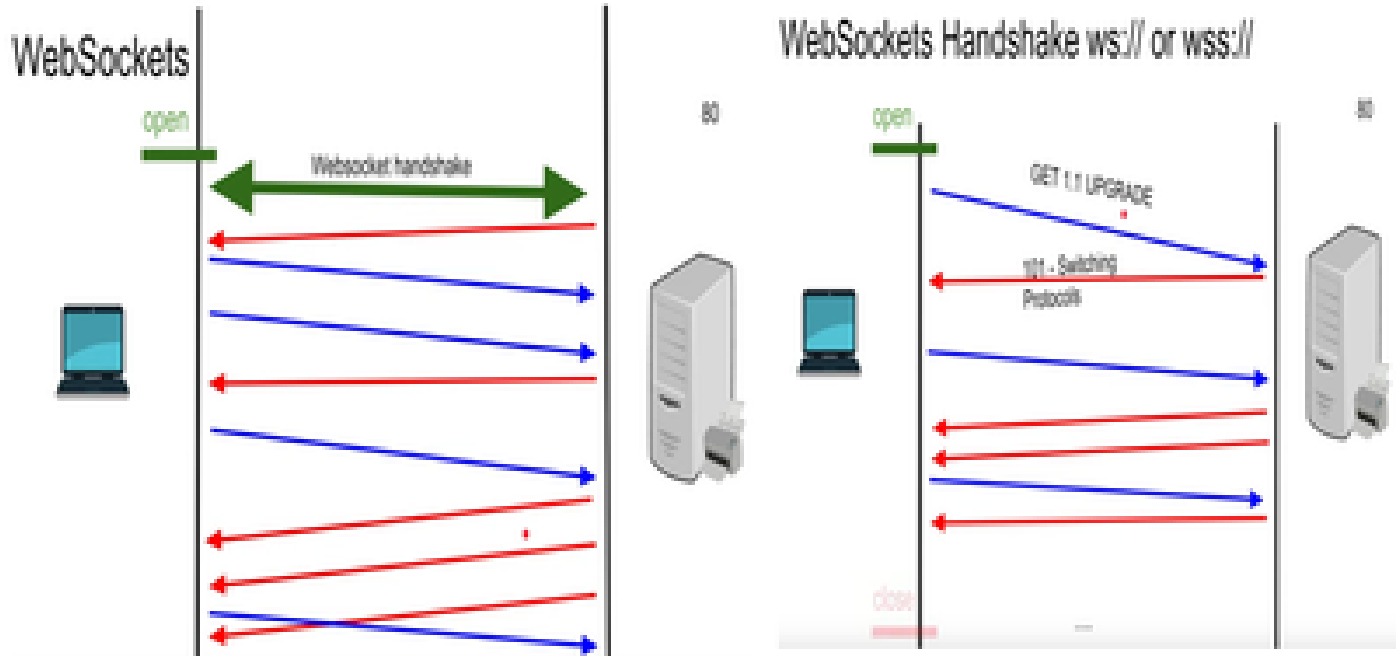
La conexión de socket web es una conexión dúplex completo bidireccional persistente.

Puede tratarse de una llamada telefónica entre un servidor y un cliente en la que cualquier persona puede hablar con la otra en cualquier momento.

Acciones de WebSocket

1. Para establecer una conexión de socket web, el cliente envía una solicitud de protocolo de enlace HTTP con un encabezado actualizado o actualizado.
    1. Esto significa que el cliente le está diciendo al servidor que en este momento, esto es a través de HTTP pero de ahora en adelante, se mueve a la conexión de websocket.
2. El servidor responde entonces con la respuesta HTTP 101, lo que significa que el sitio está conmutando la respuesta del protocolo.
3. Después de esto, se establece la conexión de websocket.

Ahora el servidor y el cliente pueden utilizar la misma conexión para transferirse datos entre sí en cualquier momento.



Depuraciones de WebSocket

Si en este punto ingresa al cliente Finesse y ve las depuraciones de red, se muestra como:

MÉTODO: GET
Dominio - NOMBRE DEL SERVIDOR

ARCHIVO - /WS/

INICIADOR - Openfire.js - websocket

Examinando solicitud y respuesta:

Petición

GET

Esquema: wss

host: uccxpub.prabhat.com:8445

nombre de archivo: /ws/

Dirección: IP del servidor uccx

Estado: 101

Switching Protocolos

VersiónHTTP/1.1

ENCABEZADO DE RESPUESTA

Conexión: actualizar

Actualizar: WebSocket

Request – open

Response

PLAIN

http://jabber.org/protocol/caps" hash="sha-1" node="
https://www.igniterealtime.org/projects/openfire/" ver="k3mOuil8afx3OTZxYy6yxLmFsok="/>

Request - auth

YWRtaW5pc3RyYXRvckB1Y2N4cHViLnByYWJoYXQuY29tAGFkbWluaXN0cmF0b3IAMTIzNA==

Response

Request – XMPP Bind Bind request to Bind the resource which in this case is desktop with a jabber id

desktop

Response – XMPP Bind where User ID is given a jabber id

administrator@uccxpub.prabhat.com/desktop

administrator@uccxpub.prabhat.com/desktop

Presence request

Presence response

http://jabber.org/protocol/caps" hash="sha-1" node="
http://www.igniterealtime.org/projects/smack" ver="NfJ3flI83zSdUDzCEICtbypursw=">

http://jabber.org/protocol/caps" hash="sha-1" node="
http://www.igniterealtime.org/projects/smack" ver="NfJ3flI83zSdUDzCEICtbypursw=">

PUBSUB request – Requesting to subscribe the user to the pubsub node so that all the events on the user are monitored.

Response – user subscribed.

http://jabber.org/protocol/pubsub">

PUBSUB request – Requesting to subscribe the Team to the pubsub node so that all the events on the team are monitored.

Response – Team subscribed

http://jabber.org/protocol/pubsub">

Información Relacionada

- Soporte técnico y descargas de Cisco

Acerca de esta traducción
Cisco ha traducido este documento combinando la traducción automática y los recursos humanos a fin de ofrecer a nuestros usuarios en todo el mundo contenido en su propio idioma.

Tenga en cuenta que incluso la mejor traducción automática podría no ser tan precisa como la proporcionada por un traductor profesional.

Cisco Systems, Inc. no asume ninguna responsabilidad por la precisión de estas traducciones y recomienda remitirse siempre al documento original escrito en inglés (insertar vínculo URL).