WebSocket Connection für Finesse verstehen

Inhalt

Einleitung

In diesem Dokument wird die WebSocket-Verbindung vollständig beschrieben, sodass bei der Fehlerbehebung die zugrunde liegenden Prozesse genau verstanden werden.

Voraussetzungen

Anforderungen

Es gibt keine spezifischen Anforderungen für dieses Dokument.

Komponenten Verwendet

Die Informationen in diesem Dokument basierend auf folgenden Software- und Hardware-Versionen:

- Cisco Finesse
- UCCX

Die Informationen in diesem Dokument beziehen sich auf Geräte in einer speziell eingerichteten Testumgebung. Alle Geräte, die in diesem Dokument benutzt wurden, begannen mit einer gelöschten (Nichterfüllungs) Konfiguration. Wenn Ihr Netzwerk in Betrieb ist, stellen Sie sicher, dass Sie die möglichen Auswirkungen aller Befehle kennen.

Hintergrundinformationen

Web Socket ist eine permanente Verbindung zwischen dem Client und dem Server.

Web-Socket

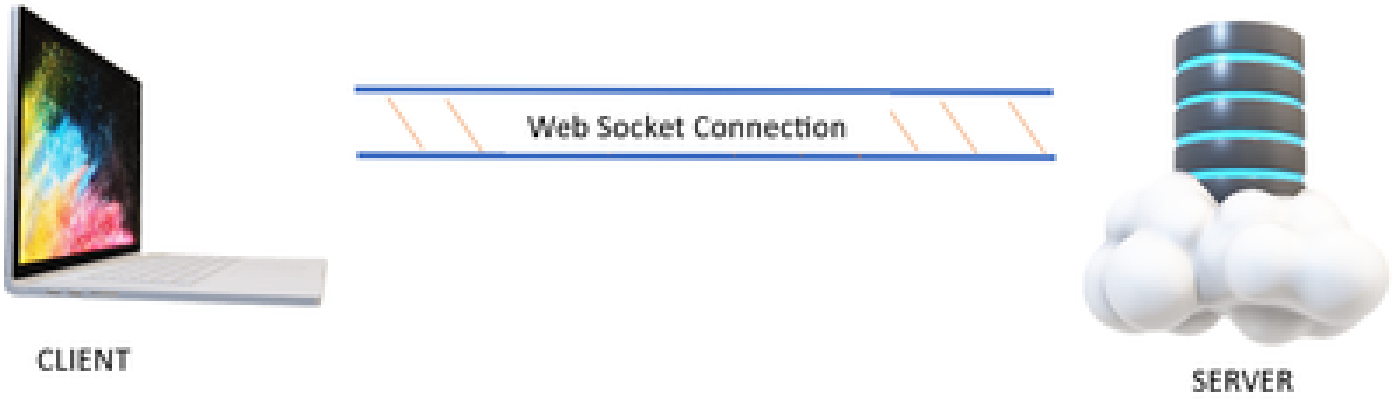
Was versteht man unter permanenter Verbindung?

Das bedeutet, dass Client und Server nach dem Herstellen der Verbindung zwischen Client und Server jederzeit Daten senden und/oder empfangen können.

Dies ist eine bidirektionale Vollduplexverbindung.

Der Server muss nicht warten, bis die Client-Anforderung Daten zurücksendet.

Ebenso muss der Client nicht jedes Mal eine neue Verbindung herstellen, um neue Daten an den Server zu senden.

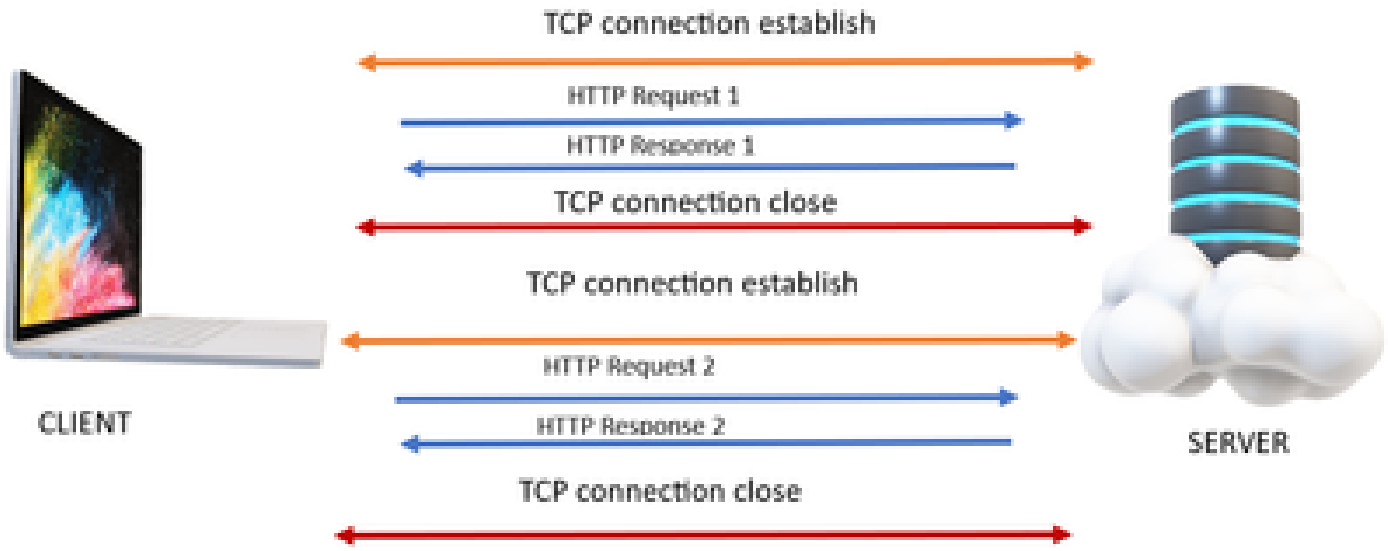


Die Web Socket-Verbindung wird hauptsächlich in Anwendungen verwendet, in denen Echtzeitdaten aktualisiert werden müssen.

Beispielsweise Anwendungen für den Aktienhandel, Messaging-Anwendungen und in unserem Fall Cisco Finesse.

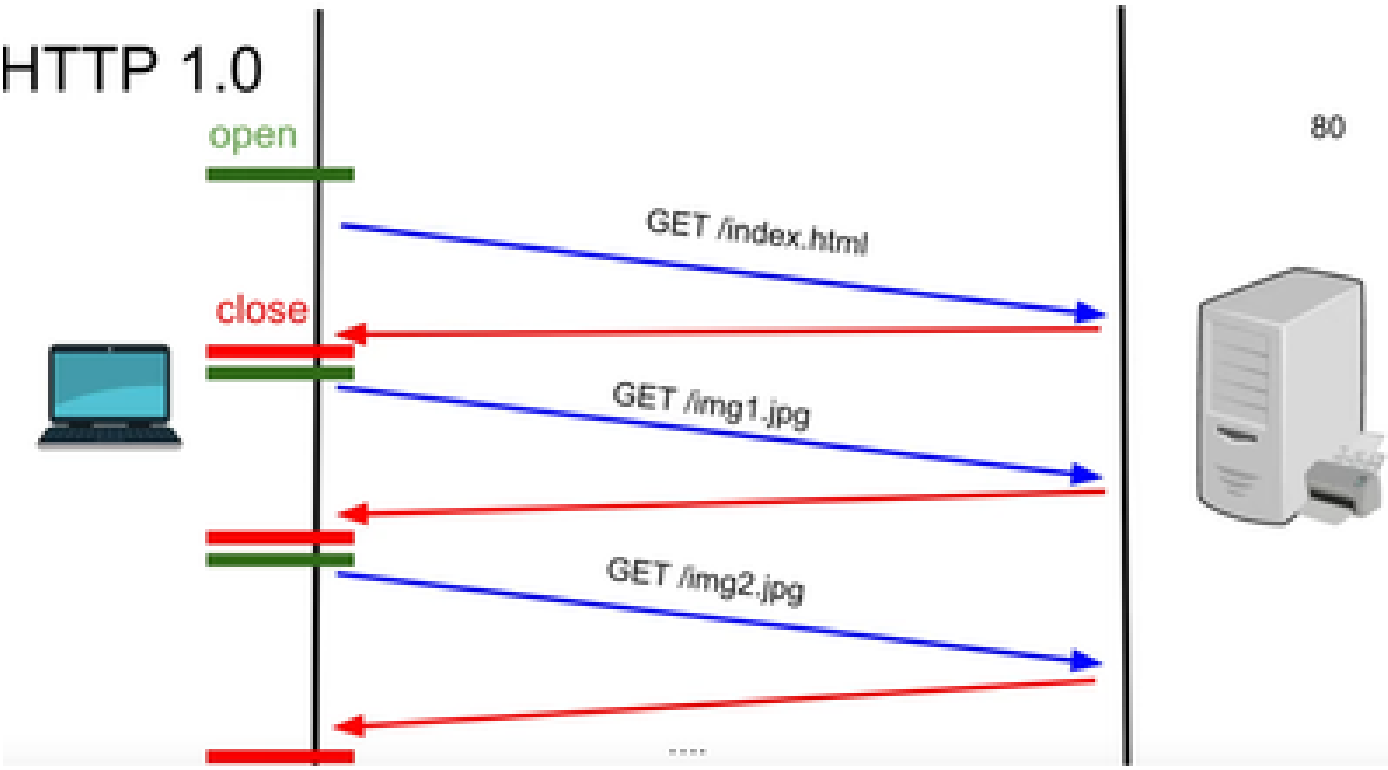
Wie funktionieren WebSockets?

Erwägung:

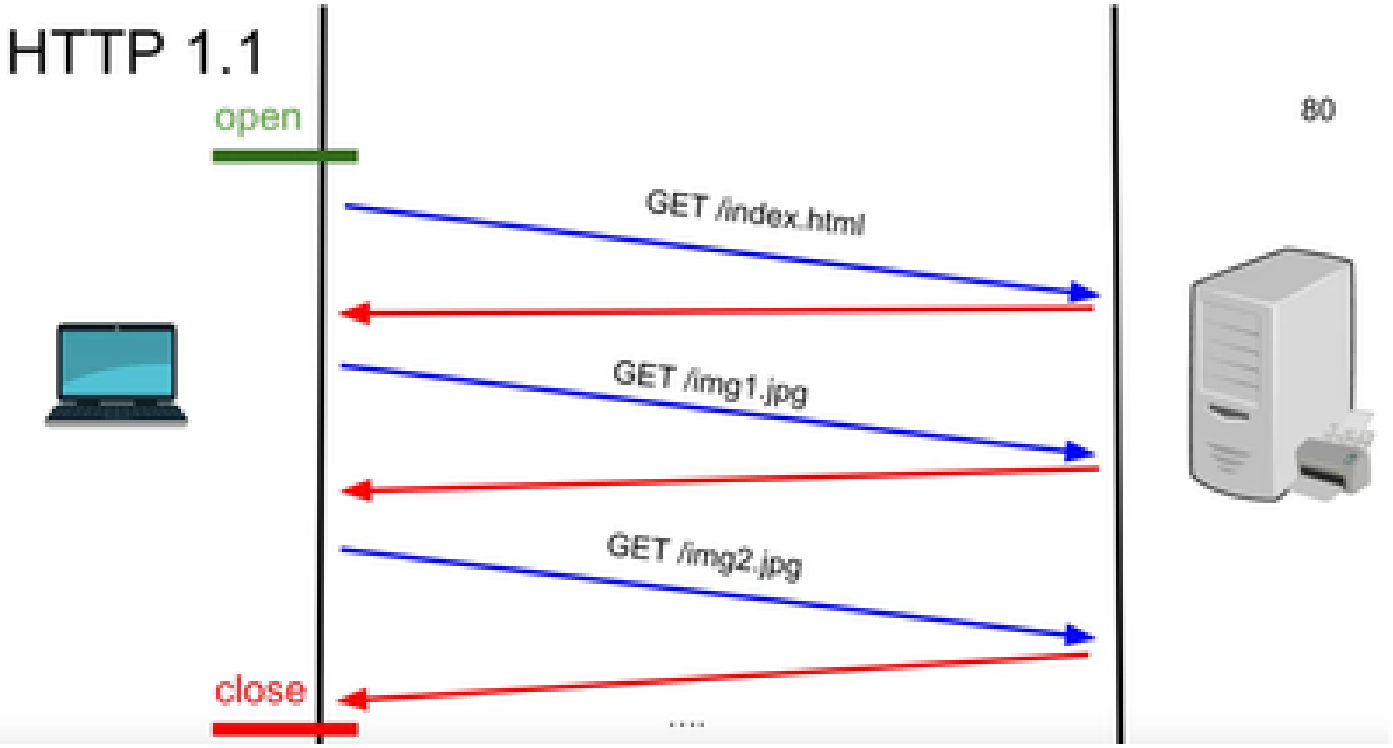


HTTP

1. Die TCP-Verbindung (3-Wege-Handshake) erfolgt.
2. Anschließend sendet der Client eine HTTP-Anfrage.
3. Der Server sendet eine HTTP-Antwort.
4. Nach einem Anforderungsantwortzyklus wird die TCP-Verbindung geschlossen.
5. Bei einer neuen HTTP-Anforderung wird wiederum zuerst die TCP-Verbindung hergestellt.



HTTP 1.0 - Nach jeder Anforderungsantwort startet der TCP-Handshake für eine andere HTTP-Anforderungsantwort erneut.



HTTP 1.1 - Diese Verbindung funktionierte, weil Sie Daten senden und empfangen und dann die Verbindung schließen konnten.

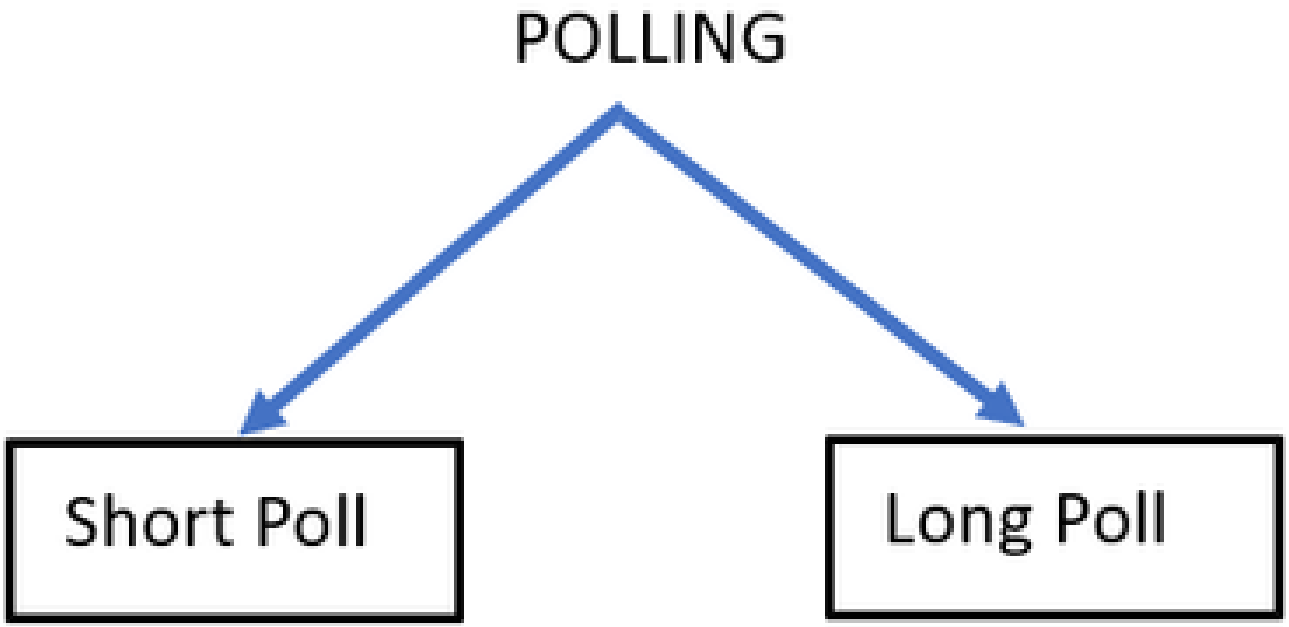
Auch dies war nicht für Echtzeit-Apps geeignet, da der Server einige Daten senden kann, auch wenn der Client sie nicht anfordert. Daher ist dieses Modell nicht wirksam.

Problem mit HTTP

Das Problem beginnt bei den Echtzeitsystemen.

Für eine Website, die Echtzeit-Updates benötigt, ist es sehr schwierig, HTTP-Anfragen jedes Mal zu senden, um ein Update vom Server zu erhalten, und es verwendet eine Menge Bandbreite und verursacht Überlastung.

Um dieses Problem zu lösen, wird ein HTTP-Polling-Mechanismus verwendet.



Short Poll (Kurzabfrage) - Das wird implementiert, wenn ein kurzer fester Zeitgeber für die Anfragen und Antworten festgelegt wird. Beispiel: 0,5 Sekunden oder 1 Sekunde, je nach Implementierung.

Wenn es kein Update von der anderen Seite, dann können Sie leere Antworten in diesem Zeitrahmen erhalten, die Ressourcen verschwenden können.

Lange Umfrage - Sie überwindet die kurze Umfrage irgendwie, hat aber immer noch eine feste Zeit, auf eine Antwort zu warten.

Wenn es keine Antwort in diesem Zeitrahmen, die relativ länger als kurze Umfrage, aber immer noch fest ist, dann wieder Anfrage Timeouts.

Daher ist Polling nicht der beste Weg, um dieses Problem zu überwinden.

Die andere zu verwendende Methode ist SSE.

SSE

Server hat Ereignisse gesendet

Dabei besteht eine unidirektionale Verbindung zwischen dem Server und dem Client, über die der Server die Daten jederzeit an den Client senden kann.

Hierbei ist zu beachten, dass es sich um eine unidirektionale Verbindung handelt, d. h. dass nur der Server die Daten an den Client senden kann und nicht umgekehrt.

Ein Beispiel für einen Anwendungsfall ist: Massenbenachrichtigungen oder -aktualisierungen von einem Server an einen Client. Zum Beispiel Nachrichten Live-Updates, Instagram Live, und so weiter.

Dies ist bei Anwendungen mit Echtzeit-Updates und Messaging nicht sehr effektiv.

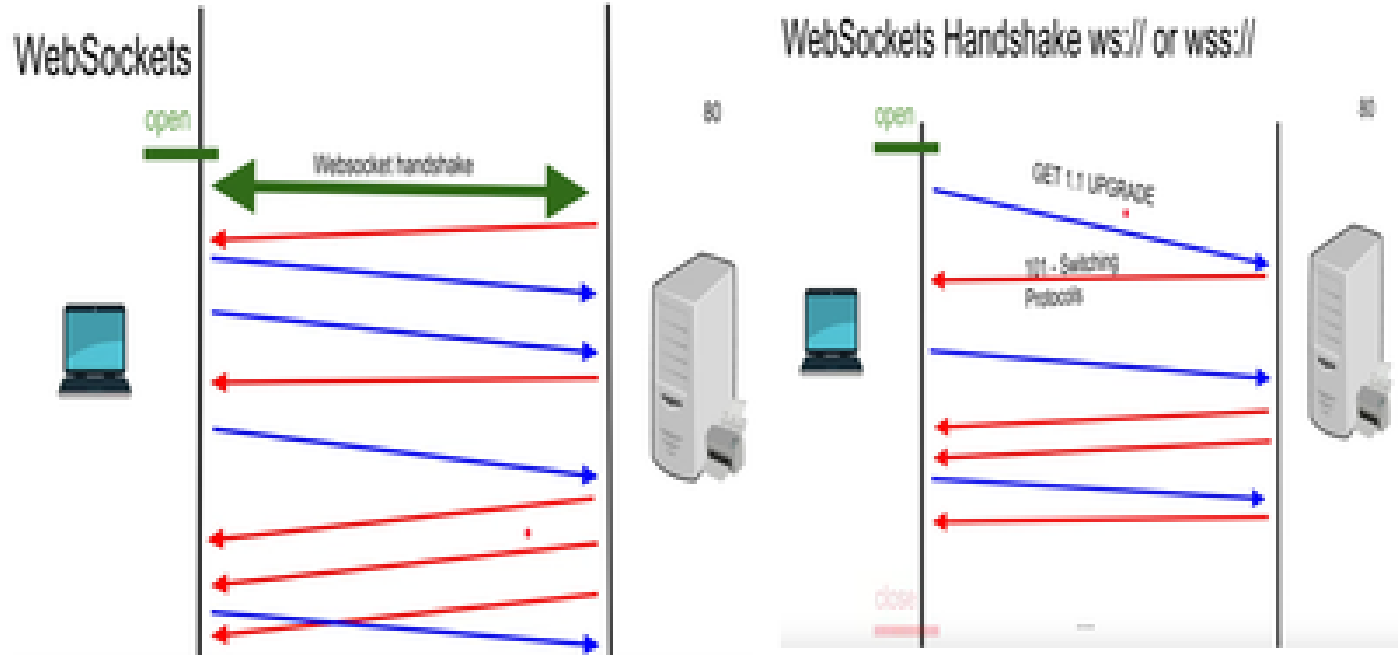
Die Web Socket Connection ist eine permanente bidirektionale Vollduplex-Verbindung.

Hierbei kann es sich um einen Telefonanruf zwischen einem Server und einem Client handeln, bei dem jeder Teilnehmer jederzeit mit dem anderen Teilnehmer kommunizieren kann.

WebSocket-Aktionen

1. Um eine Websocket-Verbindung herzustellen, sendet der Client eine HTTP-Handshake-Anforderung mit einem aktualisierten oder aktualisierten Header.
    1. Das bedeutet, dass der Client dem Server mitteilt, dass dies momentan über HTTP geschieht, aber von nun an auf die Websocket-Verbindung übergeht.
2. Der Server antwortet dann mit der HTTP 101-Antwort, was bedeutet, dass die Site die Protokollantwort umschaltet.
3. Danach wird die Websocket-Verbindung hergestellt.

Jetzt können Server und Client über dieselbe Verbindung jederzeit Daten untereinander übertragen.



WebSocket-Debugs

Wenn Sie sich an dieser Stelle beim Finesse-Client anmelden und die Netzwerkdebugs sehen, wird Folgendes angezeigt:

METHODE - GET
Domäne - SERVERNAME

DATEI - /WS/

INITIATOR - Openfire.js - Websocket

Prüfung der Anfrage und Antwort:

Anfrage

HOLEN

Schema: wss

Host: uccxpub.prabhat.com:8445

Dateiname: /ws/

Adresse: IP des UCCX-Servers

Status: 101

Switching Protokolle

VersionHTTP/1.1

ANTWORT-HEADER

Verbindung: Upgrade

Upgrade: 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">

Zugehörige Informationen

- Technischer Support und Downloads von Cisco

Informationen zu dieser Übersetzung

Cisco hat dieses Dokument maschinell übersetzen und von einem menschlichen Übersetzer editieren und korrigieren lassen, um unseren Benutzern auf der ganzen Welt Support-Inhalte in ihrer eigenen Sprache zu bieten. Bitte beachten Sie, dass selbst die beste maschinelle Übersetzung nicht so genau ist wie eine von einem professionellen Übersetzer angefertigte. Cisco Systems, Inc. übernimmt keine Haftung für die Richtigkeit dieser Übersetzungen und empfiehlt, immer das englische Originaldokument (siehe bereitgestellter Link) heranzuziehen.