Entender o WebSocket Connection para Finesse

Contents

Introdução

Este documento descreve a conexão WebSocket completamente para que, durante a solução de problemas, os processos subjacentes sejam completamente entendidos.

Pré-requisitos

Requisitos

Não existem requisitos específicos para este documento.

Componentes Utilizado

As informações neste documento são baseadas nestas versões de software e hardware:

- Cisco Finesse
- UCCX

As informações neste documento foram criadas a partir de dispositivos em um ambiente de laboratório específico. Todos os dispositivos utilizados neste documento foram iniciados com uma configuração (padrão) inicial. Se a rede estiver ativa, certifique-se de que você entenda o impacto potencial de qualquer comando.

Informações de Apoio

O Web Socket é uma conexão persistente entre o cliente e o servidor.

Soquete da Web

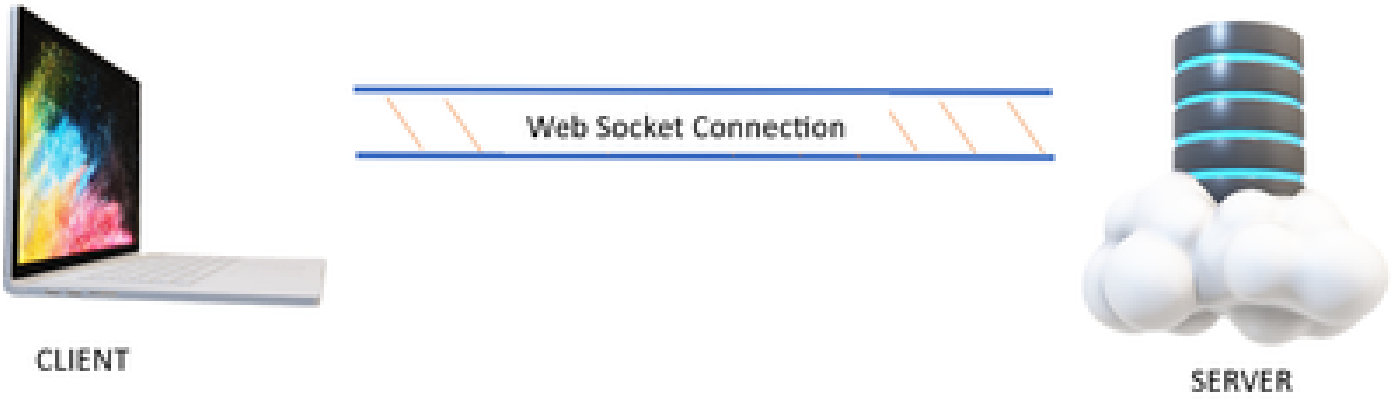
O que significa o termo conexão persistente?

Isso significa que, uma vez estabelecida a conexão entre o cliente e o servidor, o cliente e o servidor podem enviar e/ou receber dados a qualquer momento.

Esta é uma conexão bidirecional full-duplex.

O servidor não precisa aguardar a solicitação do cliente para enviar quaisquer dados.

Da mesma forma, o cliente também não precisa criar uma nova conexão toda vez para enviar novos dados ao servidor.

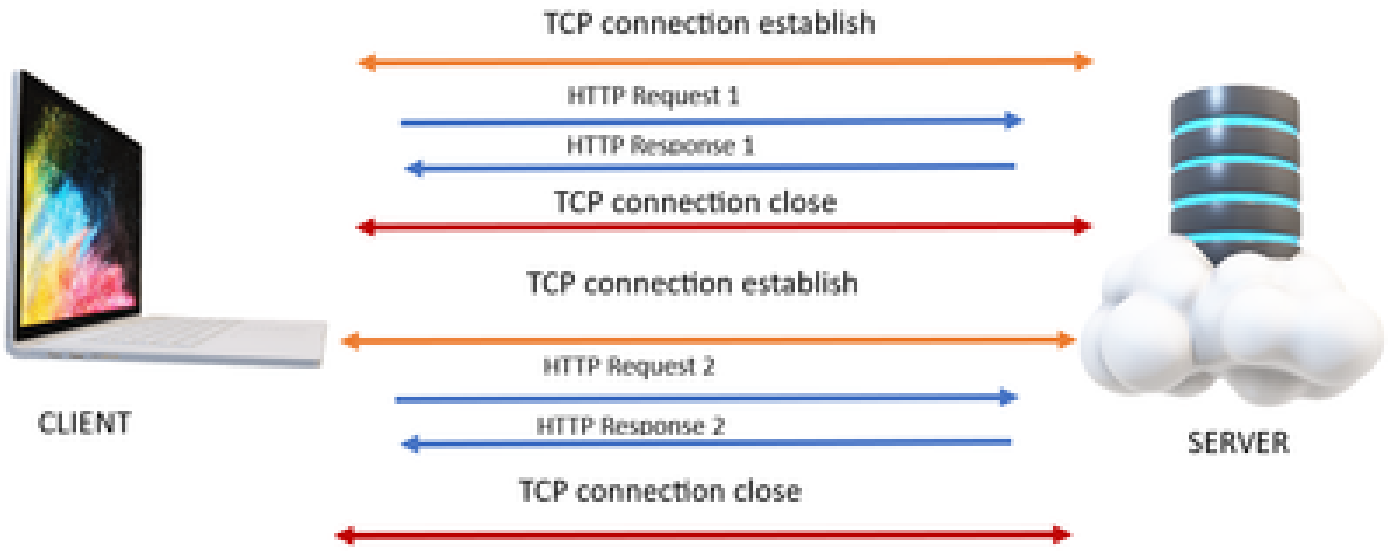


A conexão com o Web Socket é usada principalmente nos aplicativos em que são necessárias atualizações de dados em tempo real.

Por exemplo, aplicativos de troca de ações, aplicativos de mensagens e, no nosso caso, Cisco Finesse.

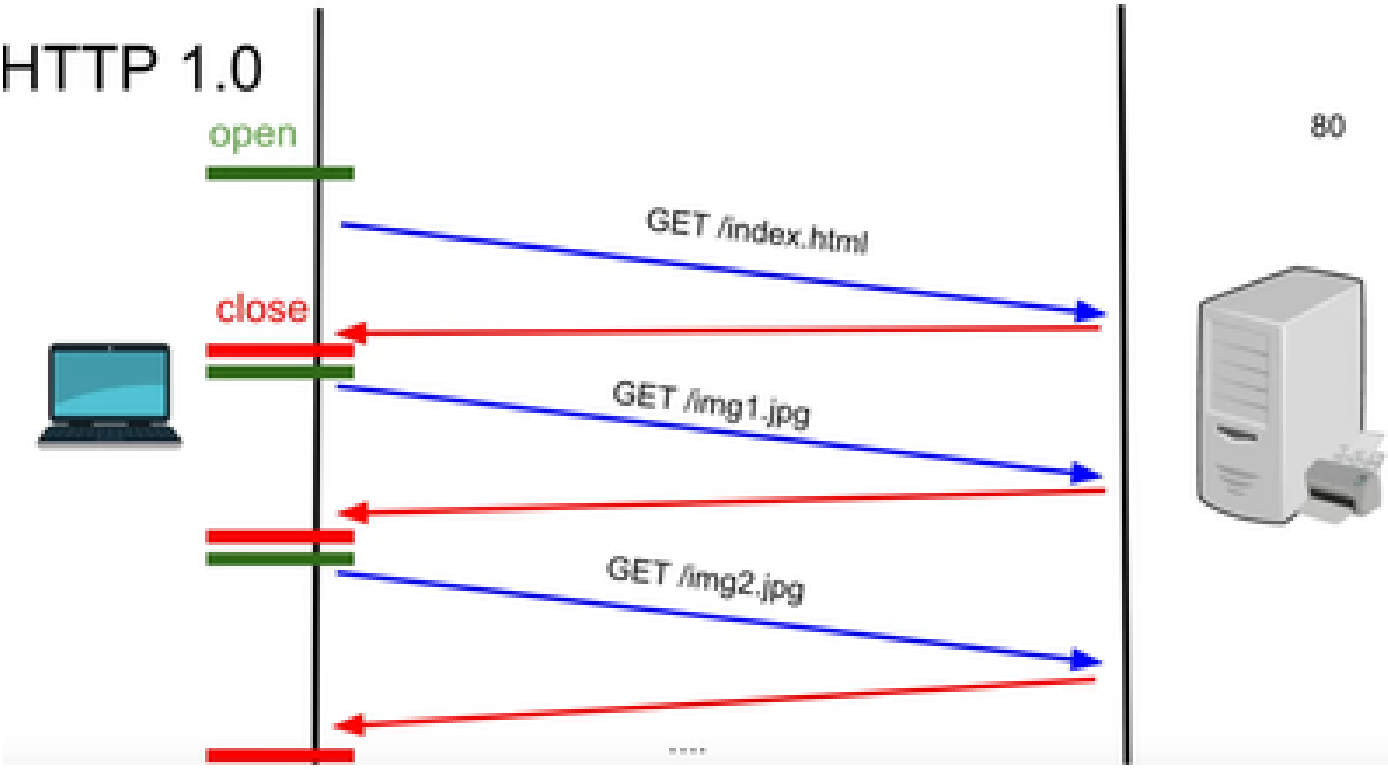
Como funcionam os WebSockets?

Considere:

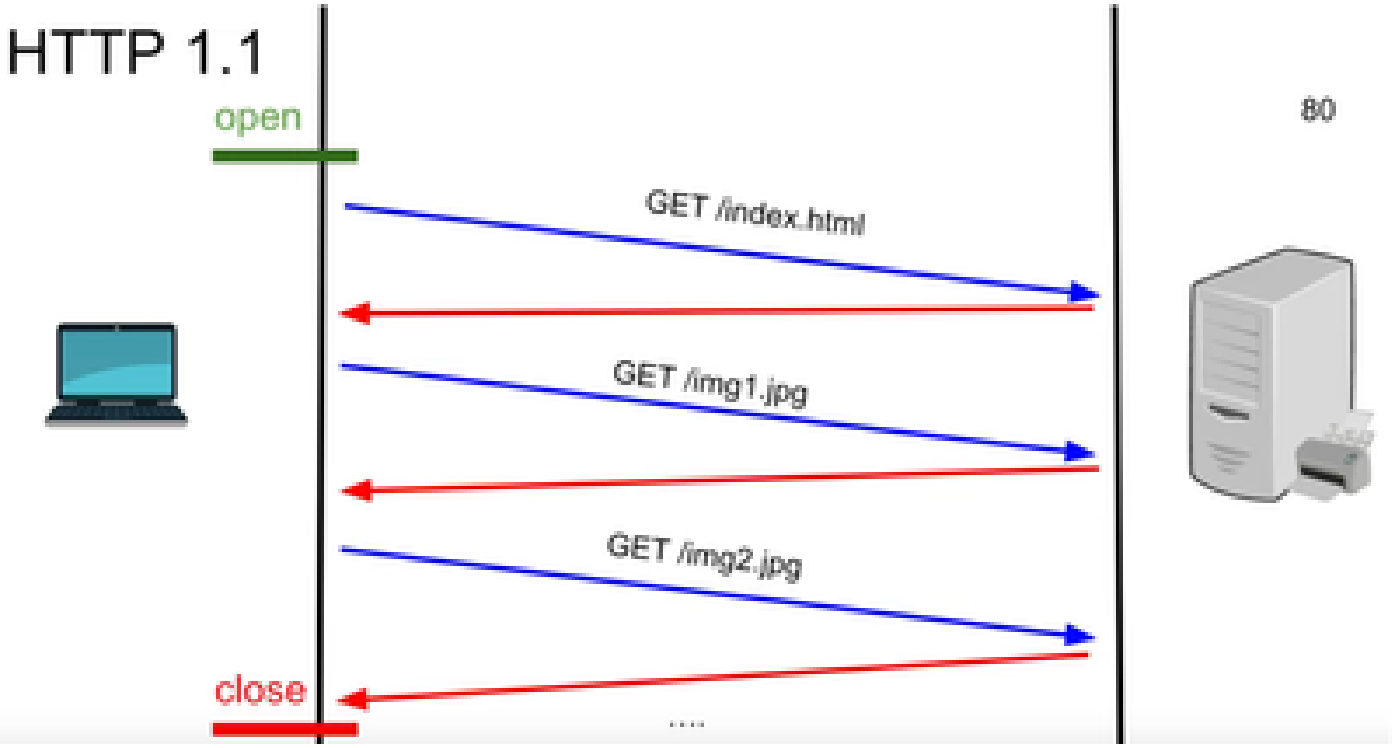


HTTP

1. A conexão TCP (handshake triplo) ocorre.
2. Em seguida, o cliente envia a solicitação HTTP.
3. O servidor envia resposta HTTP.
4. Após um ciclo de resposta de solicitação, a conexão TCP é fechada.
5. Para uma nova solicitação HTTP, novamente, a conexão TCP é estabelecida primeiro.



HTTP 1.0 - Após cada resposta de solicitação, o handshake TCP é iniciado novamente para outra resposta de solicitação HTTP.



HTTP 1.1 - Esta conexão funcionou porque você poderia enviar e receber dados e depois fechar a conexão.

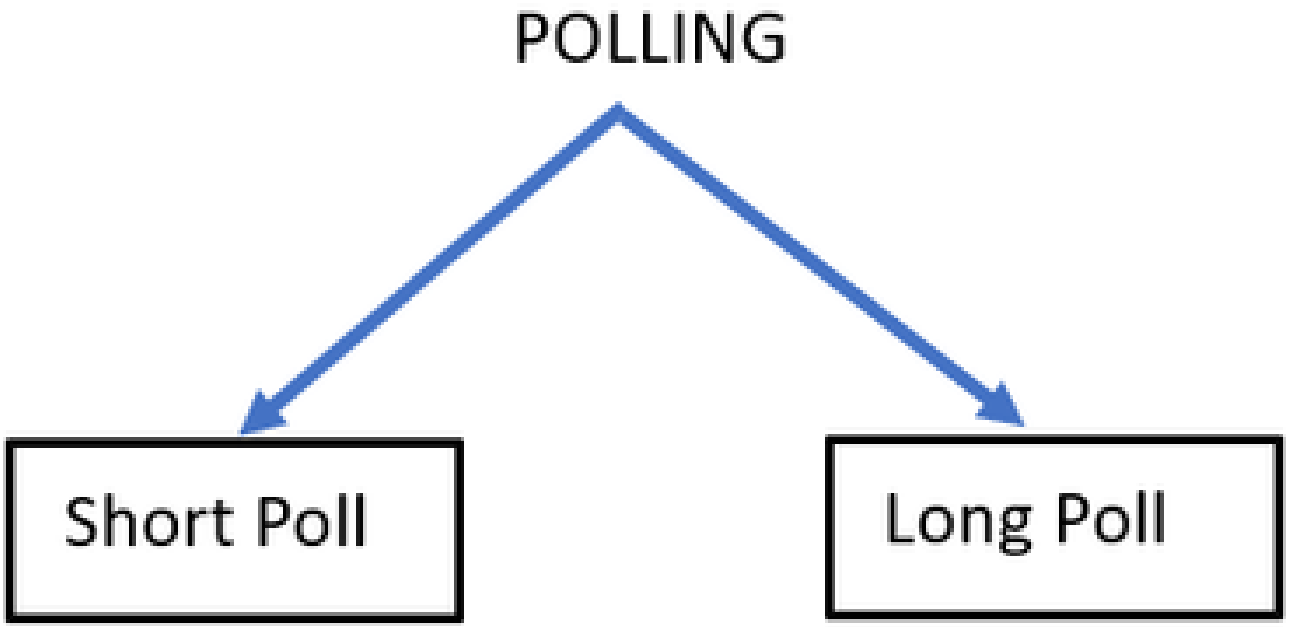
Novamente, isso não era adequado para aplicativos em tempo real porque o servidor pode enviar alguns dados mesmo quando o cliente não está solicitando. Portanto, esse modelo não é eficaz.

Problema com HTTP

O problema começa com os sistemas em tempo real.

Para um site que requer atualizações em tempo real, é muito difícil enviar solicitações HTTP todas as vezes para obter uma atualização do servidor e ele usa muita largura de banda e causa sobrecarga.

Para resolver isso, um mecanismo de HTTP chamado Polling é usado.



Pesquisa curta - A é implementada quando um temporizador fixo curto é definido para as solicitações e respostas. Por exemplo, 05 segundos ou 1 segundo dependendo da implementação.

Se não houver nenhuma atualização do outro lado, você poderá obter respostas vazias nesse período de tempo que pode desperdiçar recursos.

Pesquisa longa - Ela de alguma forma supera a pesquisa curta, mas ainda tem um tempo fixo para esperar por uma resposta.

Se não houver resposta nesse intervalo de tempo que seja relativamente maior que uma pesquisa curta, mas ainda assim ela for corrigida, solicite novamente intervalos.

Portanto, a pesquisa não é a melhor maneira de superar esse problema.

Para isso, o outro método a ser usado é chamado SSE.

SSE

Eventos enviados do servidor

Nesse caso, há uma conexão unidirecional entre o servidor e o cliente pela qual o servidor pode enviar os dados ao cliente em qualquer ponto.

O que deve ser observado aqui é que é uma conexão unidirecional, o que significa que somente o servidor pode enviar os dados ao cliente e não o contrário.

Um exemplo de caso de uso é: notificações em massa ou atualizações de um servidor para um cliente. Por exemplo, notícias ao vivo, Instagram ao vivo, etc.

Isso não é muito eficaz para aplicativos que envolvem atualizações e mensagens em tempo real.

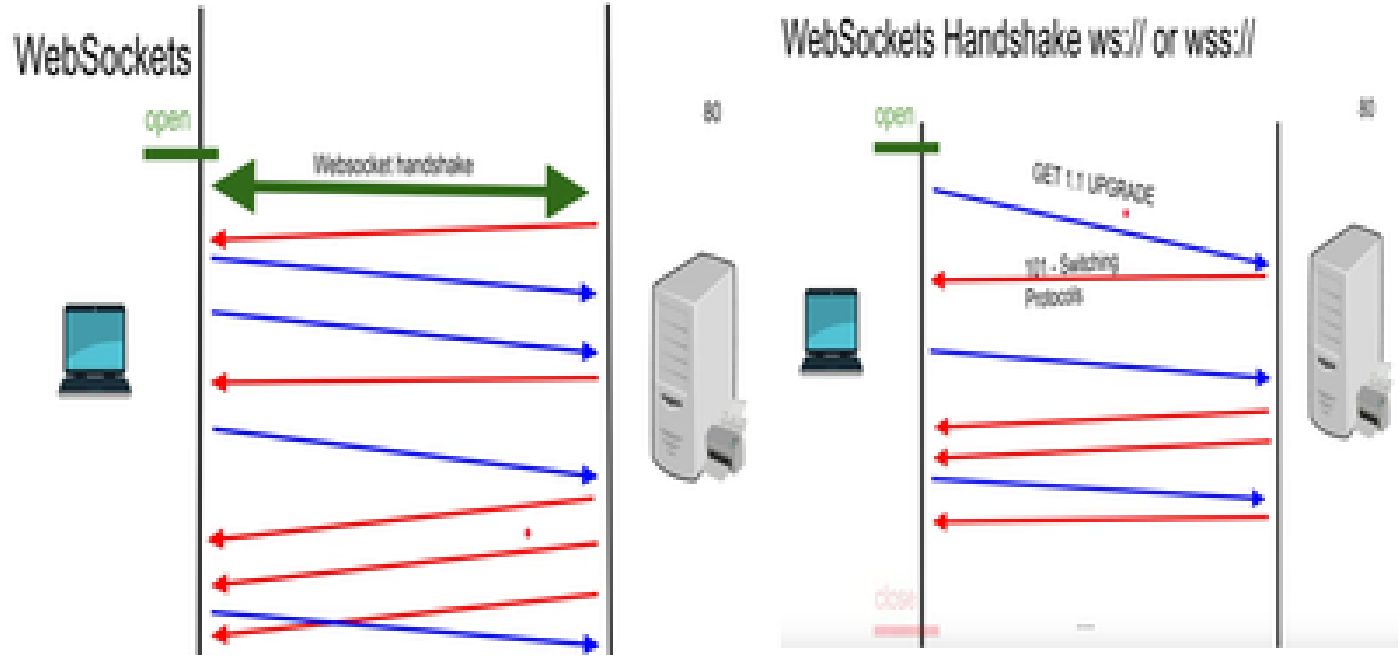
A Conexão de Soquete da Web é uma conexão full-duplex bidirecional persistente.

Pode ser uma chamada telefônica entre um servidor e um cliente em que qualquer parte pode falar com a outra parte a qualquer momento.

Ações do WebSocket

1. Para estabelecer uma conexão de websocket, o cliente envia uma solicitação de handshake HTTP com um cabeçalho atualizado ou atualizado.
    1. Isso significa que o cliente está dizendo ao servidor que agora, isso é por HTTP, mas a partir de agora, ele passa para a conexão de websocket.
2. O servidor responde com uma resposta HTTP 101, o que significa que o site está alternando a resposta do protocolo.
3. Depois disso, a conexão do websocket é estabelecida.

Agora, o servidor e o cliente podem usar a mesma conexão para transferir dados entre si a qualquer momento.



Depurações WebSocket

Se neste ponto você fizer login no cliente Finesse e ver as depurações de rede, ele será exibido como:

MÉTODO - GET
Domínio - NOME DO SERVIDOR

ARQUIVO - /WS/

INICIADOR - Openfire.js - websocket

Examinando solicitação e resposta:

Requisição

GET

Esquema: wss

host: uccxpub.prabhat.com:8445

nome de arquivo : /ws/

Endereço: IP do servidor uccx

Status: 101

Comutação Protocolos

VersãoHTTP/1.1

CABEÇALHO DE RESPOSTA

Conexão: atualização

Atualização: 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">

Informações Relacionadas

- Suporte técnico e downloads da Cisco

Sobre esta tradução
A Cisco traduziu este documento com a ajuda de tecnologias de tradução automática e humana para oferecer conteúdo de suporte aos seus usuários no seu próprio idioma, independentemente da localização.

Observe que mesmo a melhor tradução automática não será tão precisa quanto as realizadas por um tradutor profissional.

A Cisco Systems, Inc. não se responsabiliza pela precisão destas traduções e recomenda que o documento original em inglês (link fornecido) seja sempre consultado.